

# Theoretische Informatik 2

Johannes Köbler



Institut für Informatik  
Humboldt-Universität zu Berlin

WS 2009/10

# Termine

## Vorlesung

- Di 09-11 RUD 25 3.001 J. Köbler
- Do 09-11 RUD 25 3.001 J. Köbler

## Übungen

- Di 11-13 RUD 26 1'307 S. Tazari
- Di 11-13 RUD 25 3.101 W. Kössler
- Di 15-17 RUD 26 1'305 S. Tazari
- Do 11-13 RUD 26 1'307 S. Kuhnert
- Fr 09-11 RUD 25 3.113 W. Kössler

Weitere Infos unter [www.informatik.hu-berlin.de/forschung/gebiete/algorithmenII/Lehre/ws09/theo2](http://www.informatik.hu-berlin.de/forschung/gebiete/algorithmenII/Lehre/ws09/theo2)

# Übungen (Anmeldung über GOYA erforderlich)

## Abgabe der Aufgabenblätter

- in der VL, auf GOYA und der VL-Webseite

## Bearbeitung

- in Gruppen von maximal **drei** Teilnehmern
- bitte Namen und Matrikelnr. angeben

## Abgabe

- bis **9:10 Uhr** im Erdgeschoss von Haus 4!  
(Abgabetermin s. Aufgabenblatt)
- bitte nur in Papierform

## Rückgabe

- in den Übungsgruppen

# Schein, Klausur, Skript

## Scheinkriterien

- Lösen von  $\geq 50\%$  der schriftlichen Aufgaben,
- Erfolgreiches Vorrechnen von  $\geq 3$  mündl. Aufgaben.

## Klausur

- **Termin: 23.02.2010**
- **Zulassung** nur mit Übungsschein
- Nachklausur: nach dem SS 2010

## Skript

- wird wöchentlich ins Netz gestellt.

Gibt es zum organisatorischen Ablauf noch Fragen?

# Inhalt der Vorlesung

## Zentrale Themen von Th1 1:

- Mathem. Grundlagen der Informatik, Beweise führen, Modellierung  
Aussagenlogik, Prädikatenlogik
- Welche Probleme sind lösbar?  
Berechenbarkeitstheorie

## Zentrale Themen von Th1 2:

- Welche Rechenmodelle sind adäquat? **Automatentheorie**
- Welcher Aufwand ist nötig? **Komplexitätstheorie**

## Zentrales Thema von Th1 3:

- Wie lassen sich praktisch relevante Problemstellungen möglichst effizient lösen? **Algorithmik**

# Maschinenmodelle

- Rechenmaschinen spielen in der Informatik eine zentrale Rolle.
- Je nach Berechnungskraft unterschiedliche math. Modelle.
- In Th11: TM als ein universales Berechnungsmodell.
- In Th13: Registermaschine (engl. random access machine; RAM).
- Dieses Modell ist etwas flexibler als die TM, da es den unmittelbaren Lese- und Schreibzugriff (**random access**) auf eine beliebige Speichereinheit (Register) erlaubt.
- Hier betrachten wir Einschränkungen des TM-Modells, die vielfältige praktische Anwendungen haben, wie z.B.
  - endliche Automaten (DFA, NFA),
  - Kellerautomaten (PDA, DPDA) etc.

# Der Algorithmenbegriff

- Der Begriff **Algorithmus** geht auf den persischen Gelehrten **Muhammed Al Chwarizmi** (8./9. Jhd.) zurück.
- Ältester bekannter nicht-trivialer Algorithmus: **Euklidischer Algorithmus** zur Berechnung des größten gemeinsamen Teilers zweier natürlicher Zahlen (300 v. Chr.).
- Von einem Algorithmus wird erwartet, dass er bei jeder zulässigen **Problemeingabe** nach endlich vielen Rechenschritten eine korrekte **Ausgabe** liefert.
- Ein Algorithmus ist ein „Verfahren“ zur Lösung eines Berechnungsproblems, das sich prinzipiell auf einer Turingmaschine implementieren lässt (**Church-Turing-These**).
- Problemeingaben können Zahlen, Formeln, Graphen etc. sein.
- Diese werden über einem Eingabealphabet  $\Sigma$  kodiert.



# Alphabet, Wort, Sprache

## Definition

- Ein **Alphabet** ist eine geordnete endliche Menge

$$\Sigma = \{a_1, \dots, a_m\}, \quad m \geq 1$$

von **Zeichen**  $a_i$ .

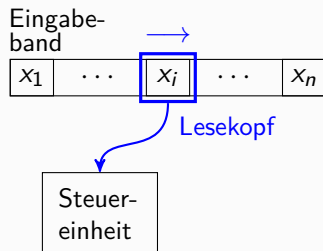
- Eine Folge  $x = x_1 \dots x_n \in \Sigma^n$  heißt **Wort** (der **Länge**  $n$ ).
- Die Menge aller Wörter über  $\Sigma$  ist

$$\Sigma^* = \bigcup_{n \geq 0} \Sigma^n.$$

- Das (einzige) Wort der Länge  $n = 0$  ist das **leere Wort**, welches wir mit  $\varepsilon$  bezeichnen.
- Jede Teilmenge  $L \subseteq \Sigma^*$  heißt **Sprache** über dem Alphabet  $\Sigma$ .

# Das Rechenmodell des endlichen Automaten

## Ein endlicher Automat



- ist eine „abgespeckte“ Turingmaschine,
- verfügt nur über konstant viel Speicherplatz,
- macht bei Eingaben der Länge  $n$  nur  $n$  Rechenschritte und
- liest in jedem Schritt genau ein Eingabezeichen.

# Formale Definition eines endlichen Automaten

## Definition

Ein **endlicher Automat** (kurz: **DFA**; *deterministic finite automaton*) wird durch ein 5-Tupel  $M = (Z, \Sigma, \delta, q_0, E)$  beschrieben, wobei

- $Z \neq \emptyset$  eine **endliche** Menge von **Zuständen**,
- $\Sigma$  das **Eingabealphabet**,
- $\delta : Z \times \Sigma \rightarrow Z$  die **Überföhrungsfunktion**,
- $q_0 \in Z$  der **Startzustand** und
- $E \subseteq Z$  die Menge der **Endzustände** ist.

Die von  $M$  **akzeptierte** oder **erkannte Sprache** ist

$$L(M) = \left\{ x_1 \cdots x_n \in \Sigma^* \mid \begin{array}{l} \exists q_1, \dots, q_{n-1} \in Z, q_n \in E: \\ \delta(q_i, x_{i+1}) = q_{i+1} \text{ f\u00fcr } i = 0, \dots, n-1 \end{array} \right\}.$$

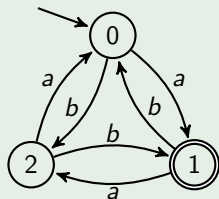
# DFAs beherrschen Modulare Arithmetik

## Beispiel

Sei  $M_3 = (Z, \Sigma, \delta, 0, E)$  ein DFA mit  $Z = \{0, 1, 2\}$ ,  
 $\Sigma = \{a, b\}$ ,  $E = \{1\}$  und

$\delta$	0	1	2
a	1	2	0
b	2	0	1

Graphische  
Darstellung:



Endzustände werden durch einen doppelten Kreis und der Startzustand durch einen Pfeil gekennzeichnet. <

## Behauptung

Die von  $M_3$  erkannte Sprache ist

$$L(M_3) = \{x \in \{a, b\}^* \mid \#_a(x) - \#_b(x) \equiv_3 1\}, \text{ wobei}$$

- $\#_a(x)$  die Anzahl der Vorkommen von  $a$  in  $x$  bezeichnet und
- $i \equiv_m j$  bedeutet, dass  $i - j$  durch  $m$  teilbar ist.

# Das Erreichbarkeitsproblem für DFAs

## Behauptung

Die von  $M_3$  erkannte Sprache ist  $\{x \in \{a, b\}^* \mid \#_a(x) - \#_b(x) \equiv_3 1\}$ .

## Beweis der Behauptung durch Induktion über die Länge von $x$

Wir betrachten zunächst das Erreichbarkeitsproblem für DFAs.

## Frage

Sei  $M = (Z, \Sigma, \delta, q_0, E)$  ein DFA und sei  $x = x_1 \cdots x_n \in \Sigma^*$ . Welchen Zustand erreicht  $M$  bei Eingabe  $x$  nach  $i$  Schritten?

## Antwort

- nach 0 Schritten:  $q_0$ ,
- nach 1 Schritt:  $\delta(q_0, x_1)$ ,
- nach 2 Schritten:  $\delta(\delta(q_0, x_1), x_2)$ ,
- nach  $i$  Schritten:  $\delta(\dots \delta(\delta(q_0, x_1), x_2), \dots x_i)$ .

# Das Erreichbarkeitsproblem für DFAs

## Definition

- Bezeichne  $\hat{\delta}(q, x)$  denjenigen Zustand, in dem sich  $M$  nach Lesen von  $x$  befindet, wenn  $M$  im Zustand  $q$  gestartet wird.
- Dann können wir die Funktion

$$\hat{\delta} : Z \times \Sigma^* \rightarrow Z$$

induktiv (über die Länge von  $x$ ) wie folgt definieren.

- Für  $q \in Z$ ,  $x \in \Sigma^*$  und  $a \in \Sigma$  sei

$$\begin{aligned}\hat{\delta}(q, \varepsilon) &= q, \\ \hat{\delta}(q, xa) &= \delta(\hat{\delta}(q, x), a).\end{aligned}$$

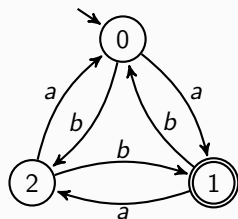
- Die von  $M$  erkannte Sprache lässt sich nun auch in der Form

$$L(M) = \{x \in \Sigma^* \mid \hat{\delta}(q_0, x) \in E\}$$

schreiben.

# DFAs beherrschen Modulare Arithmetik

$M_3$



Behauptung

$$L(M_3) = \{x \in \{a, b\}^* \mid \#_a(x) - \#_b(x) \equiv_3 1\}.$$

Beweis

- 1 ist der einzige Endzustand von  $M$ .
- Daher ist  $L(M_3) = \{x \in \Sigma^* \mid \hat{\delta}(0, x) = 1\}$ .
- Folglich reicht es, folgende Kongruenzgleichung zu zeigen:

$$\hat{\delta}(0, x) \equiv_3 \#_a(x) - \#_b(x)$$

# DFAs beherrschen Modulare Arithmetik

Beweis von  $\hat{\delta}(0, x) \equiv_3 \#_a(x) - \#_b(x)$ :

Wir führen Induktion über die Länge  $n$  von  $x$ .

**Induktionsanfang  $n = 0$ :** klar, da  $\hat{\delta}(0, \varepsilon) = \#_a(\varepsilon) = \#_b(\varepsilon) = 0$  ist.

**Induktionsschritt  $n \rightsquigarrow n + 1$ :**

- Sei  $x = x_1 \cdots x_{n+1}$  gegeben und sei  $i = \hat{\delta}(0, x_1 \cdots x_n)$ .
- Nach IV ist

$$i \equiv_3 \#_a(x_1 \cdots x_n) - \#_b(x_1 \cdots x_n).$$

- Wegen  $\delta(i, a) \equiv_3 i + 1$  und  $\delta(i, b) \equiv_3 i - 1$  folgt

$$\delta(i, x_{n+1}) \equiv_3 i + \#_a(x_{n+1}) - \#_b(x_{n+1}) = \#_a(x) - \#_b(x).$$

- Folglich ist

$$\hat{\delta}(0, x) = \delta(\hat{\delta}(0, x_1 \cdots x_n), x_{n+1}) = \delta(i, x_{n+1}) \equiv_3 \#_a(x) - \#_b(x).$$





# Die Klasse der regulären Sprachen

## Definition

Eine von einem DFA akzeptierte Sprache wird als **regulär** bezeichnet. Die zugehörige Sprachklasse ist

$$\text{REG} = \{L(M) \mid M \text{ ist ein DFA}\}.$$

## Frage

Welche Sprachen gehören zu REG und welche nicht?

# Singletons sind regulär

## Vereinbarung

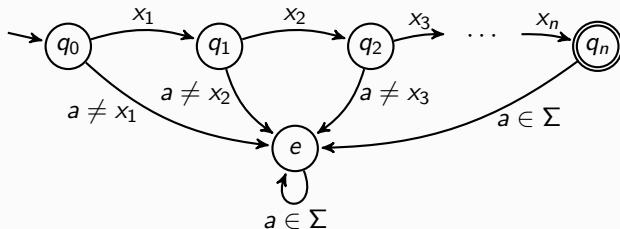
Für das Folgende sei  $\Sigma = \{a_1, \dots, a_m\}$  ein fest gewähltes Alphabet.

## Beobachtung 1

Alle Sprachen, die nur ein Wort  $x = x_1 \cdots x_n \in \Sigma^*$  enthalten, sind regulär.

## Beweis

Folgender DFA  $M$  erkennt die Sprache  $L(M) = \{x\}$ :



# Abschlusseigenschaften von Sprachklassen

## Definition

Ein ( $k$ -stelliger) Sprachoperator ist eine Abbildung  $op$ , die  $k$  Sprachen  $L_1, \dots, L_k$  auf eine Sprache  $op(L_1, \dots, L_k)$  abbildet.

## Beispiel

Der 2-stellige Schnittoperator bildet zwei Sprachen  $L_1$  und  $L_2$  auf die Sprache  $L_1 \cap L_2$  ab.



## Definition

- Eine Sprachklasse  $\mathcal{K}$  heißt unter  $op$  abgeschlossen, wenn gilt:

$$L_1, \dots, L_k \in \mathcal{K} \Rightarrow op(L_1, \dots, L_k) \in \mathcal{K}.$$

- Der Abschluss von  $\mathcal{K}$  unter  $op$  ist die kleinste Sprachklasse  $\mathcal{K}'$ , die  $\mathcal{K}$  enthält und unter  $op$  abgeschlossen ist.

# Reguläre Sprachen sind unter Komplement abgeschlossen

## Beobachtung 2

Ist  $L \in \text{REG}$ , so ist auch die Sprache  $\bar{L} = \Sigma^* \setminus L$  regulär.

## Beweis

- Sei  $M = (Z, \Sigma, \delta, q_0, E)$  ein DFA mit  $L(M) = L$ .
- Dann akzeptiert der DFA

$$\bar{M} = (Z, \Sigma, \delta, q_0, Z \setminus E)$$

das Komplement  $\bar{L}$  von  $L$ .



# Reguläre Sprachen sind unter Durchschnitt abgeschlossen

## Beobachtung 3

Sind  $L_1, L_2 \in \text{REG}$ , so ist auch die Sprache  $L_1 \cap L_2$  regulär.

## Beweis

- Seien  $M_i = (Z_i, \Sigma, \delta_i, q_i, E_i)$ ,  $i = 1, 2$ , DFAs mit  $L(M_i) = L_i$ .
- Dann wird der Schnitt  $L_1 \cap L_2$  von dem DFA

$$M = (Z_1 \times Z_2, \Sigma, \delta, (q_1, q_2), E_1 \times E_2)$$

mit

$$\delta((p, q), a) = (\delta_1(p, a), \delta_2(q, a))$$

erkannt.



# Reguläre Sprachen sind unter Vereinigung abgeschlossen

## Beobachtung 4

Die Vereinigung  $L_1 \cup L_2$  von regulären Sprachen  $L_1$  und  $L_2$  ist regulär.

## Beweis

Es gilt  $L_1 \cup L_2 = \overline{\overline{L_1} \cap \overline{L_2}}$ . □

## Frage

Wie sieht der zugehörige DFA aus?

## Antwort

$$M' = (Z_1 \times Z_2, \Sigma, \delta, (q_1, q_2), (E_1 \times Z_2) \cup (Z_1 \times E_2)).$$

# REG ist unter Mengenoperationen abgeschlossen

## Korollar

Die Klasse REG der regulären Sprachen ist unter folgenden Operationen abgeschlossen:

- Komplement,
- Durchschnitt,
- Vereinigung.

# Wie umfangreich ist REG?

## Folgerung

- Aus den Beobachtungen folgt, dass alle **endlichen** und alle **co-endlichen** Sprachen regulär sind.
- Da die reguläre Sprache

$$L(M_3) = \{x \in \{a, b\}^* \mid \#_a(x) - \#_b(x) \equiv_3 1\}$$

weder endlich noch co-endlich ist, haben wir damit allerdings noch nicht alle regulären Sprachen erfasst.



# Konstruktive Charakterisierung von REG

## Frage

Lassen sich alle regulären Sprachen aus endlichen Sprachen mithilfe von einfachen Operationen gewinnen?

## Definition

- Das **Produkt** (**Verkettung**, **Konkatenation**) der Sprachen  $L_1$  und  $L_2$  ist  $L_1L_2 = \{xy \mid x \in L_1, y \in L_2\}$ .
- Ist  $L_1 = \{x\}$  einelementig (diese Sprachen werden auch als **Singleton-sprachen** bezeichnet), so schreiben wir für  $\{x\}L_2$  auch einfach  $xL_2$ .
- Die  **$n$ -fache Potenz  $L^n$**  einer Sprache  $L$  ist induktiv definiert durch

$$L^n = \begin{cases} \{\varepsilon\}, & n = 0, \\ L^{n-1}L, & n > 0. \end{cases}$$

- Die **Sternhülle** von  $L$  ist  $L^* = \bigcup_{n \geq 0} L^n$ .
- Die **Plushülle** von  $L$  ist  $L^+ = \bigcup_{n \geq 1} L^n = LL^*$ .

# Überblick

## Ziel

Zeige, dass REG als Abschluss der endl. Sprachen unter Vereinigung, Produktbildung und Sternhülle charakterisierbar ist.

## Problem

Bei der Konstruktion eines DFA für das Produkt  $L_1L_2$  bereitet es Schwierigkeiten, den richtigen Zeitpunkt für den Übergang von (der Simulation von)  $M_1$  zu  $M_2$  zu finden.

## Lösungsidee

Ein **nichtdeterministischer** Automat (NFA) kann den richtigen Zeitpunkt für den Übergang „erraten“.

## Verbleibendes Problem

Zeige, dass auch NFAs nur reguläre Sprachen erkennen.

# Nichtdeterministische Automaten

## Definition

- Ein **nichtdet. endl. Automat** (kurz: **NFA**; *nondet. finite automaton*)

$$N = (Z, \Sigma, \delta, Q_0, E)$$

ist genau so aufgebaut wie ein DFA, nur dass er

- eine Menge  $Q_0 \subseteq Z$  von Startzuständen hat und
- die Überföhrungsfunktion folgende Form hat:

$$\delta : Z \times \Sigma \rightarrow \mathcal{P}(Z).$$

- Hierbei bezeichnet  $\mathcal{P}(Z)$  die **Potenzmenge** (also die Menge aller Teilmengen) von  $Z$ . Diese wird auch oft mit  $2^Z$  bezeichnet.
- Die von einem NFA  $N$  **akzeptierte Sprache** ist

$$L(N) = \left\{ x_1 \cdots x_n \in \Sigma^* \mid \exists q_0 \in Q_0, q_1, \dots, q_{n-1} \in Z, q_n \in E: \right. \\ \left. q_{i+1} \in \delta(q_i, x_{i+1}) \text{ f\u00fcr } i = 0, \dots, n-1 \right\}.$$

# Eigenschaften von NFAs

- Ein NFA  $N$  kann also nicht nur eine, sondern mehrere verschiedene Rechnungen parallel ausführen.
- Die Eingabe  $x$  wird genau dann akzeptiert, wenn mind. eine Rechnung von  $N$  nach Lesen von  $x$  einen Endzustand erreicht.
- Im Gegensatz zu einem DFA, der jede Eingabe zu Ende liest, kann ein NFA  $N$  „stecken bleiben“.
- Dieser Fall tritt ein, wenn  $N$  in einen Zustand  $q$  gelangt, in dem er das nächste Eingabezeichen  $x_i$  wegen

$$\delta(q, x_i) = \emptyset$$

nicht verarbeiten kann.

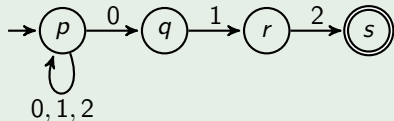
# Eigenschaften von NFAs

## Beispiel

- Betrachte den NFA  $N = (Z, \Sigma, \delta, Q_0, E)$  mit  $Z = \{p, q, r, s\}$ ,  $\Sigma = \{0, 1, 2\}$ ,  $Q_0 = \{p\}$ ,  $E = \{s\}$  und der Überföhrungsfunktion

$\delta$	$p$	$q$	$r$	$s$
0	$\{p, q\}$	$\emptyset$	$\emptyset$	$\emptyset$
1	$\{p\}$	$\{r\}$	$\emptyset$	$\emptyset$
2	$\{p\}$	$\emptyset$	$\{s\}$	$\emptyset$

Graphische Darstellung:



- Dann ist  $L(M) = \{x012 \mid x \in \Sigma^*\}$  die Sprache aller W6rter, die mit dem Suffix 012 enden.

# Eigenschaften von NFAs

## Beobachtung 5

Seien  $N_i = (Z_i, \Sigma, \delta_i, Q_i, E_i)$  NFAs mit  $L(N_i) = L_i$  für  $i = 1, 2$ . Dann wird auch das Produkt  $L_1 L_2$  von einem NFA erkannt.

## Beweis

- Wir können  $Z_1 \cap Z_2 = \emptyset$  annehmen.
- Die Sprache  $L_1 L_2$  wird dann von dem NFA  $N = (Z_1 \cup Z_2, \Sigma, \delta, Q_1, E)$  mit

$$\delta(p, a) = \begin{cases} \delta_1(p, a), & p \in Z_1 \setminus E_1, \\ \delta_1(p, a) \cup \bigcup_{q \in Q_2} \delta_2(q, a), & p \in E_1, \\ \delta_2(p, a), & \text{sonst} \end{cases}$$

und

$$E = \begin{cases} E_1 \cup E_2, & Q_2 \cap E_2 \neq \emptyset \\ E_2, & \text{sonst} \end{cases}$$

akzeptiert.



# Eigenschaften von NFAs

## Beobachtung 6

Ist  $N = (Z, \Sigma, \delta, Q_0, E)$  ein NFA, so wird auch die Sprache  $L(N)^*$  von einem NFA erkannt.

## Beweis

Die Sprache  $L(N)^*$  wird von dem NFA

$$N^* = (Z \cup \{q_{neu}\}, \Sigma, \delta^*, Q_0 \cup \{q_{neu}\}, E \cup \{q_{neu}\})$$

mit

$$\delta^*(p, a) = \begin{cases} \delta(p, a) \cup \bigcup_{q \in Q_0} \delta(q, a), & p \in E, \\ \delta(p, a), & \text{sonst} \end{cases}$$

erkannt. □

# Überblick

## Ziel

Zeige, dass REG als Abschluss der endl. Sprachen unter Vereinigung, Produktbildung und Sternhülle charakterisierbar ist.

## Problem

Bei der Konstruktion eines DFA für das Produkt  $L_1L_2$  bereitet es Schwierigkeiten, den richtigen Zeitpunkt für den Übergang von (der Simulation von)  $M_1$  zu  $M_2$  zu finden.

## Lösungsidee (bereits umgesetzt)

Ein **nichtdeterministischer** Automat (NFA) kann den richtigen Zeitpunkt für den Übergang „erraten“.

## Noch zu zeigen

NFAs erkennen genau die regulären Sprachen.



# NFAs erkennen genau die regulären Sprachen

Satz (Rabin und Scott)

$\text{REG} = \{L(N) \mid N \text{ ist ein NFA}\}.$

Beweis von  $\text{REG} \subseteq \{L(N) \mid N \text{ ist ein NFA}\}$

Diese Inklusion ist klar, da jeder DFA  $M = (Z, \Sigma, \delta, q_0, E)$  leicht in einen äquivalenten NFA

$$N = (Z, \Sigma, \bar{\delta}, Q_0, E)$$

transformiert werden kann, indem wir  $\bar{\delta}(q, a) = \{\delta(q, a)\}$  und  $Q_0 = \{q_0\}$  setzen. □

Für die umgekehrte Inklusion ist das **Erreichbarkeitsproblem für NFAs** von zentraler Bedeutung.

# Das Erreichbarkeitsproblem für NFAs

## Frage

Sei  $N = (Z, \Sigma, \delta, Q_0, E)$  ein NFA und sei  $x = x_1 \cdots x_n \in \Sigma^*$ . Welche Zustände kann  $N(x)$  in  $i$  Schritten erreichen?

## Antwort

- in 0 Schritten: alle Zustände in  $Q_0$ .
- in einem Schritt: alle Zustände in

$$Q_1 = \bigcup_{q \in Q_0} \delta(q, x_1).$$

- in  $i$  Schritten: alle Zustände in

$$Q_i = \bigcup_{q \in Q_{i-1}} \delta(q, x_i).$$

# Simulation von NFAs durch DFAs

## Idee

- Wir können einen NFA  $N = (Z, \Sigma, \delta, Q_0, E)$  durch einen DFA  $M$  simulieren, der in seinem Zustand die Information speichert, in welchen Zuständen sich  $N$  momentan befinden könnte.
- Die Zustände von  $M$  sind also Teilmengen  $Q$  von  $Z$  mit  $Q_0$  als Startzustand und der Endzustandsmenge

$$E' = \{Q \subseteq Z \mid Q \cap E \neq \emptyset\}.$$

- Die Überföhrungsfunktion  $\delta' : \mathcal{P}(Z) \times \Sigma \rightarrow \mathcal{P}(Z)$  von  $M$  berechnet dann für einen Zustand  $Q \subseteq Z$  und ein Zeichen  $a \in \Sigma$  die Menge

$$\delta'(Q, a) = \bigcup_{q \in Q} \delta(q, a)$$

aller Zustände, in die  $N$  gelangen kann, wenn  $N$  ausgehend von einem beliebigen Zustand  $q \in Q$  das Zeichen  $a$  liest.

- Die von  $M = (\mathcal{P}(Z), \Sigma, \delta', Q_0, E')$  erkannte Sprache ist

$$L(M) = \{x \in \Sigma^* \mid \hat{\delta}'(Q_0, x) \cap E \neq \emptyset\}.$$

# NFAs erkennen genau die regulären Sprachen

## Beweis von $\{L(N) \mid N \text{ ist ein NFA}\} \subseteq \text{REG}$

- Sei  $N = (Z, \Sigma, \delta, Q_0, E)$  ein NFA und sei  $M = (\mathcal{P}(Z), \Sigma, \delta', Q_0, E')$  der zugehörige **Potenzmengenautomat** mit  $\delta'(Q, a) = \bigcup_{q \in Q} \delta(q, a)$  und  $E' = \{Q \subseteq Z \mid Q \cap E \neq \emptyset\}$ .
- Dann folgt die Korrektheit von  $M$  leicht mittels folgender Behauptung, deren Beweis wir auf der nächsten Folie nachholen.

### Behauptung

$\hat{\delta}'(Q_0, x)$  enthält genau die von  $N$  nach Lesen von  $x$  erreichbaren Zustände.

- Für alle Wörter  $x \in \Sigma^*$  gilt
  - $x \in L(N) \iff N$  kann nach Lesen von  $x$  einen Endzustand erreichen
  - $\stackrel{\text{Beh.}}{\iff} \hat{\delta}'(Q_0, x) \cap E \neq \emptyset$
  - $\iff \hat{\delta}'(Q_0, x) \in E'$
  - $\iff x \in L(M).$



# Beweis der Behauptung

## Behauptung

$\hat{\delta}'(Q_0, x)$  enthält genau die von  $N$  nach Lesen von  $x$  erreichbaren Zustände.

## Beweis durch Induktion über die Länge $n$ von $x$

$n = 0$ : klar, da  $\hat{\delta}'(Q_0, \varepsilon) = Q_0$  ist.

$n - 1 \rightsquigarrow n$ : Sei  $x = x_1 \cdots x_n$  gegeben. Nach IV enthält

$$Q_{n-1} = \hat{\delta}'(Q_0, x_1 \cdots x_{n-1})$$

die Zustände, die  $N$  nach Lesen von  $x_1 \cdots x_{n-1}$  erreichen kann. Wegen

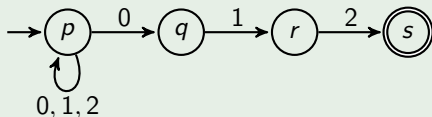
$$\hat{\delta}'(Q_0, x) = \delta'(Q_{n-1}, x_n) = \bigcup_{q \in Q_{n-1}} \delta(q, x_n)$$

enthält dann aber  $\hat{\delta}'(Q_0, x)$  die Zustände, die  $N$  nach Lesen von  $x$  erreichen kann. □

# Simulation von NFAs durch DFAs

## Beispiel

- Betrachte den NFA  $N$



mit Startzustandsmenge  $Q_0 = \{p\}$  und Endzustandsmenge  $E = \{s\}$ .

- Ausgehend von  $Q_0$  liefert  $\delta'$  dann die folgenden Werte:

$\delta'$	0	1	2
$\{p\}$	$\{p, q\}$	$\{p\}$	$\{p\}$
$\{p, q\}$	$\{p, q\}$	$\{p, r\}$	$\{p\}$
$\{p, r\}$	$\{p, q\}$	$\{p\}$	$\{p, s\}$
$\{p, s\}$	$\{p, q\}$	$\{p\}$	$\{p\}$

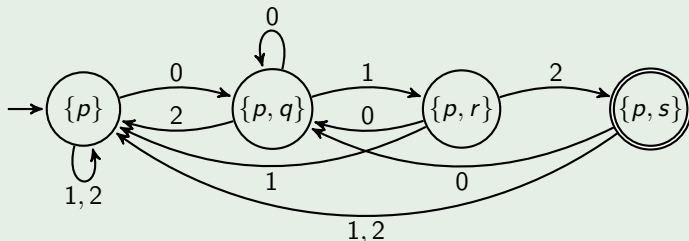
# Simulation von NFAs durch DFAs

## Beispiel

- Ausgehend von  $Q_0$  liefert  $\delta'$  dann die folgenden Werte:

$\delta'$	$\{p\}$	$\{p, q\}$	$\{p, r\}$	$\{p, s\}$
0	$\{p, q\}$	$\{p, q\}$	$\{p, q\}$	$\{p, q\}$
1	$\{p\}$	$\{p, r\}$	$\{p\}$	$\{p\}$
2	$\{p\}$	$\{p\}$	$\{p, s\}$	$\{p\}$

- Also ist  $N$  äquivalent zu folgendem **Potenzmengenautomaten**  $M$ :



# Simulation von NFAs durch DFAs

## Bemerkung

- Im obigen Beispiel werden für die Konstruktion des Potenzmengenautomaten nur 4 der insgesamt

$$\|\mathcal{P}(Z)\| = 2^{\|Z\|} = 2^4 = 16$$

Zustände benötigt, da die übrigen 12 Zustände nicht erreichbar sind.

- Es gibt jedoch Beispiele, bei denen alle  $2^{\|Z\|}$  Zustände benötigt werden (siehe Übungen).



# Abschlusseigenschaften der Klasse REG

## Korollar

Die Klasse REG der regulären Sprachen ist unter folgenden Operationen abgeschlossen:

- Komplement,
- Durchschnitt,
- Vereinigung,
- Produkt,
- Sternhülle.

# Überblick

## Nächstes Ziel

Zeige, dass REG als Abschluss der endl. Sprachen unter Vereinigung, Produkt und Sternhülle charakterisierbar ist.

## Bereits gezeigt:

Jede Sprache, die mittels der Operationen Vereinigung, Produkt und Sternhülle (sowie Durchschnitt und Komplement) angewandt auf endliche Sprachen darstellbar ist, ist regulär.

## Noch zu zeigen:

Jede reguläre Sprache lässt sich aus endlichen Sprachen mittels Vereinigung, Produkt und Sternhülle erzeugen.

# Konstruktive Charakterisierung von REG mittels regulärer Ausdrücke

## Induktive Definition der Menge $RA$ aller regulären Ausdrücke

Die Symbole  $\emptyset$ ,  $\epsilon$  und  $a$  ( $a \in \Sigma$ ) sind reguläre Ausdrücke, die

- die leere Sprache  $L(\emptyset) = \emptyset$ ,
- die Sprache  $L(\epsilon) = \{\epsilon\}$  und
- für jedes  $a \in \Sigma$  die Sprache  $L(a) = \{a\}$  beschreiben.

Sind  $\alpha$  und  $\beta$  reguläre Ausdrücke, die die Sprachen  $L(\alpha)$  und  $L(\beta)$  beschreiben, so sind auch  $\alpha\beta$ ,  $(\alpha|\beta)$  und  $(\alpha)^*$  reguläre Ausdrücke, die folgende Sprachen beschreiben:

- $L(\alpha\beta) = L(\alpha)L(\beta)$ ,
- $L(\alpha|\beta) = L(\alpha) \cup L(\beta)$ ,
- $L((\alpha)^*) = L(\alpha)^*$ .

# Reguläre Ausdrücke

## Beispiel

Die regulären Ausdrücke  $\epsilon^*$ ,  $\emptyset^*$ ,  $(0|1)^*00$  und  $(\epsilon 0|\emptyset 1^*)$  beschreiben folgende Sprachen:

$\gamma$	$\epsilon$	$\epsilon^*$	$\emptyset^*$	$(0 1)^*00$	$(\epsilon 0 \emptyset 1^*)$
$L(\gamma)$	$\{\epsilon\}$	$\{\epsilon\}$	$\{\epsilon\}$	$\{x00 \mid x \in \{0, 1\}^*\}$	$\{0\}$



## Vereinbarungen

- Um Klammern zu sparen, definieren wir folgende **Präzedenzordnung**: Der Sternoperator  $*$  bindet stärker als der Produktoperator und dieser wiederum stärker als der Vereinigungsoperator.
- Für  $((a|b(c)^*)|d)$  können wir also kurz  $a|bc^*|d$  schreiben.
- Da der reguläre Ausdruck  $\gamma\gamma^*$  die Sprache  $L(\gamma)^+$  beschreibt, verwenden wir  $\gamma^+$  als Abkürzung für den Ausdruck  $\gamma\gamma^*$ .

# Charakterisierung von REG durch reguläre Ausdrücke

## Satz

$\{L(\gamma) \mid \gamma \text{ ist ein regulärer Ausdruck}\} \subseteq \text{REG}.$

## Beweis.

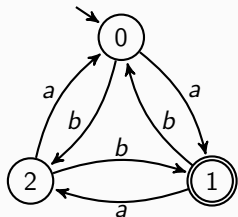
Klar, da

- die Basisausdrücke  $\emptyset$ ,  $\epsilon$  und  $a$ ,  $a \in \Sigma^*$ , nur reguläre Sprachen beschreiben und
- die Sprachklasse REG unter Produkt, Vereinigung und Sternhülle abgeschlossen ist.



# Charakterisierung von REG durch reguläre Ausdrücke

$M_3$ :



Frage

Wie lässt sich die Sprache

$$L(M_3) = \{x \in \{a, b\}^* \mid \#_a(x) - \#_b(x) \equiv_3 1\}$$

durch einen regulären Ausdruck beschreiben?

Antwort

- Die Sprache  $L_0 = \{x \in \{a, b\}^* \mid \#_a(x) - \#_b(x) \equiv_3 0\}$  lässt sich durch folgenden regulären Ausdruck beschreiben:

$$\gamma_0 = (a(ab)^*(aa|b) \mid b(ba)^*(a|bb))^*$$

- Also ist  $L(M_3)$  durch folgenden regulären Ausdruck beschreibbar:

$$\gamma_1 = \gamma_0(a|bb)(ab)^*$$

# Charakterisierung von REG durch reguläre Ausdrücke

## Satz

$\text{REG} \subseteq \{L(\gamma) \mid \gamma \text{ ist ein regulärer Ausdruck}\}.$

## Beweis

- Wir konstruieren zu einem DFA  $M = (Z, \Sigma, \delta, q_0, E)$  einen regulären Ausdruck  $\gamma$  mit  $L(\gamma) = L(M)$ .
- Wir nehmen an, dass  $Z = \{1, \dots, m\}$  und  $q_0 = 1$  ist.
- Dann lässt sich  $L(M)$  als Vereinigung

$$L(M) = \bigcup_{q \in E} L_{1,q}$$

von Sprachen der Form  $L_{p,q} = \{x \in \Sigma^* \mid \hat{\delta}(p, x) = q\}$  darstellen.

- Es reicht also, reguläre Ausdrücke für die Sprachen  $L_{p,q}$  mit  $1 \leq p, q \leq m$  anzugeben.

# Charakterisierung von REG durch reguläre Ausdrücke

## Satz

$\text{REG} \subseteq \{L(\gamma) \mid \gamma \text{ ist ein regulärer Ausdruck}\}.$

## Beweis (Fortsetzung)

- Es reicht also, reguläre Ausdrücke für die Sprachen  $L_{p,q}$  mit  $1 \leq p, q \leq m$  anzugeben.
- Hierzu betrachten wir für  $r = 0, \dots, m$  die Sprachen

$$L_{p,q}^r = \left\{ x \in L_{p,q} \mid \text{für } i = 1, \dots, n-1 \text{ ist } \hat{\delta}(p, x_1 \cdots x_i) \leq r \right\}.$$

- Wegen  $L_{p,q} = L_{p,q}^m$  reicht es, reguläre Ausdrücke für die Sprachen  $L_{p,q}^r$  mit  $1 \leq p, q \leq m$  und  $0 \leq r \leq m$  anzugeben.
- Wir zeigen induktiv über  $r$ , dass die Sprachen  $L_{p,q}^r$  durch reguläre Ausdrücke beschreibbar sind.



# Charakterisierung von REG durch reguläre Ausdrücke

## Satz

$\text{REG} \subseteq \{L(\gamma) \mid \gamma \text{ ist ein regulärer Ausdruck}\}.$

## Beweis (Schluss)

- Wir zeigen induktiv über  $r$ , dass die Sprachen  $L_{p,q}^r$  durch reguläre Ausdrücke beschreibbar sind.

$r = 0$ : In diesem Fall sind die Sprachen

$$L_{p,q}^0 = \begin{cases} \{a \in \Sigma \mid \delta(p, a) = q\}, & p \neq q, \\ \{a \in \Sigma \mid \delta(p, a) = q\} \cup \{\varepsilon\}, & \text{sonst} \end{cases}$$

endlich und somit durch reguläre Ausdrücke beschreibbar.

$r \rightsquigarrow r + 1$ : Wegen

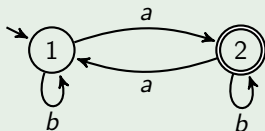
$$L_{p,q}^{r+1} = L_{p,q}^r \cup L_{p,r+1}^r (L_{r+1,r+1}^r)^* L_{r+1,q}^r$$

sind mit  $L_{p,q}^r$ ,  $1 \leq p, q \leq m$ , auch die Sprachen  $L_{p,q}^{r+1}$ ,  $1 \leq p, q \leq m$ , durch reguläre Ausdrücke beschreibbar.  $\square$

# Charakterisierung von REG durch reguläre Ausdrücke

## Beispiel

- Betrachte den DFA  $M$



- Da  $M$  insgesamt  $m = 2$  Zustände und nur den Endzustand 2 besitzt, ist

$$L(M) = \bigcup_{q \in E} L_{1,q} = L_{1,2} = L_{1,2}^2.$$

# Charakterisierung von REG durch reguläre Ausdrücke

## Beispiel (Fortsetzung)

- Um reguläre Ausdrücke  $\gamma_{p,q}^r$  für die Sprachen  $L_{p,q}^r$  zu bestimmen, benutzen wir für  $r \geq 0$  die Rekursionsformel

$$\gamma_{p,q}^{r+1} = \gamma_{p,q}^r | \gamma_{p,r+1}^r (\gamma_{r+1,r+1}^r)^* \gamma_{r+1,q}^r.$$

- Damit erhalten wir

$$\gamma_{1,2}^2 = \gamma_{1,2}^1 | \gamma_{1,2}^1 (\gamma_{2,2}^1)^* \gamma_{2,2}^1,$$

$$\gamma_{1,2}^1 = \gamma_{1,2}^0 | \gamma_{1,1}^0 (\gamma_{1,1}^0)^* \gamma_{1,2}^0,$$

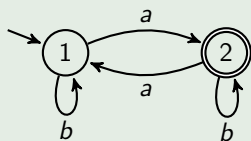
$$\gamma_{2,2}^1 = \gamma_{2,2}^0 | \gamma_{2,1}^0 (\gamma_{1,1}^0)^* \gamma_{1,2}^0.$$

- Es genügt also, die regulären Ausdrücke  $\gamma_{1,1}^0$ ,  $\gamma_{1,2}^0$ ,  $\gamma_{2,1}^0$ ,  $\gamma_{2,2}^0$ ,  $\gamma_{1,2}^1$ ,  $\gamma_{2,2}^1$  und  $\gamma_{1,2}^2$  zu berechnen.

# Charakterisierung von REG durch reguläre Ausdrücke

## Beispiel (Fortsetzung)

### DFA $M$



### Rekursionsformeln

$$L_{p,p}^0 = \{a \mid \delta(p, a) = p\} \cup \{\varepsilon\},$$

$$L_{p,q}^0 = \{a \mid \delta(p, a) = q\} \text{ für } p \neq q,$$

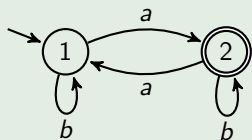
$$L_{p,q}^{r+1} = L_{p,q}^r \cup L_{p,r+1}^r (L_{r+1,r+1}^r)^* L_{r+1,q}^r.$$

$r$	$(p, q)$			
	(1, 1)	(1, 2)	(2, 1)	(2, 2)
0				
1				
2				

# Charakterisierung von REG durch reguläre Ausdrücke

## Beispiel (Fortsetzung)

DFA  $M$



Rekursionsformel

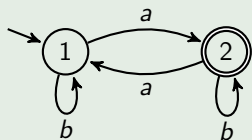
$$L_{1,1}^0 = \{a \in \Sigma \mid \delta(1, a) = 1\} \cup \{\varepsilon\} = \{\varepsilon, b\}$$

$r$	$(p, q)$			
	(1, 1)	(1, 2)	(2, 1)	(2, 2)
0				
1				
2				

# Charakterisierung von REG durch reguläre Ausdrücke

## Beispiel (Fortsetzung)

DFA  $M$



Rekursionsformel

$$L_{1,1}^0 = \{a \in \Sigma \mid \delta(1, a) = 1\} \cup \{\epsilon\} = \{\epsilon, b\}$$

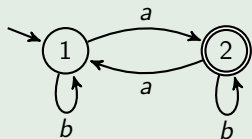
$$\rightsquigarrow \gamma_{1,1}^0 = \epsilon|b$$

$r$	$(p, q)$			
	(1, 1)	(1, 2)	(2, 1)	(2, 2)
0	$\epsilon b$			
1				
2				

# Charakterisierung von REG durch reguläre Ausdrücke

## Beispiel (Fortsetzung)

### DFA $M$



### Rekursionsformel

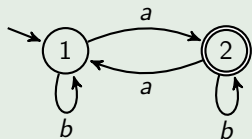
$$L_{1,2}^0 = \{a \in \Sigma \mid \delta(1, a) = 2\} = \{a\}$$

$r$	$(p, q)$			
	(1, 1)	(1, 2)	(2, 1)	(2, 2)
0	$\epsilon b$			
1				
2				

# Charakterisierung von REG durch reguläre Ausdrücke

## Beispiel (Fortsetzung)

### DFA $M$



### Rekursionsformel

$$L_{1,2}^0 = \{a \in \Sigma \mid \delta(1, a) = 2\} = \{a\}$$

$$\rightsquigarrow \gamma_{1,2}^0 = a$$

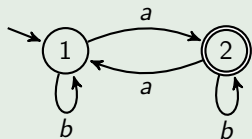
$r$	$(p, q)$			
	(1, 1)	(1, 2)	(2, 1)	(2, 2)
0	$\epsilon b$	$a$		
1				
2				



# Charakterisierung von REG durch reguläre Ausdrücke

## Beispiel (Fortsetzung)

DFA  $M$



Rekursionsformel

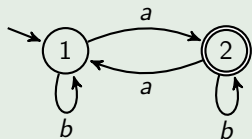
$$L_{2,1}^0 = \{a \in \Sigma \mid \delta(2, a) = 1\} = \{a\}$$

$r$	$(p, q)$			
	(1, 1)	(1, 2)	(2, 1)	(2, 2)
0	$\epsilon b$	$a$		
1				
2				

# Charakterisierung von REG durch reguläre Ausdrücke

## Beispiel (Fortsetzung)

### DFA $M$



### Rekursionsformel

$$L_{2,1}^0 = \{a \in \Sigma \mid \delta(2, a) = 1\} = \{a\}$$

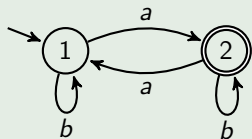
$$\rightsquigarrow \gamma_{2,1}^0 = a$$

$r$	$(p, q)$			
	(1, 1)	(1, 2)	(2, 1)	(2, 2)
0	$\epsilon b$	$a$	$a$	
1				
2				

# Charakterisierung von REG durch reguläre Ausdrücke

## Beispiel (Fortsetzung)

DFA  $M$



Rekursionsformel

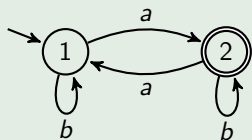
$$L_{2,2}^0 = \{a \in \Sigma \mid \delta(2, a) = 2\} \cup \{\epsilon\} = \{\epsilon, b\}$$

$r$	$(p, q)$			
	(1, 1)	(1, 2)	(2, 1)	(2, 2)
0	$\epsilon b$	$a$	$a$	
1				
2				

# Charakterisierung von REG durch reguläre Ausdrücke

## Beispiel (Fortsetzung)

DFA  $M$



Rekursionsformel

$$L_{2,2}^0 = \{a \in \Sigma \mid \delta(2, a) = 2\} \cup \{\epsilon\} = \{\epsilon, b\}$$

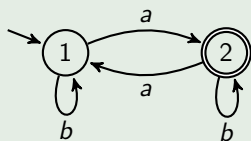
$$\rightsquigarrow \gamma_{2,2}^0 = \epsilon|b$$

$r$	$(p, q)$			
	(1, 1)	(1, 2)	(2, 1)	(2, 2)
0	$\epsilon b$	$a$	$a$	$\epsilon b$
1	-			
2				

# Charakterisierung von REG durch reguläre Ausdrücke

## Beispiel (Fortsetzung)

DFA  $M$



Rekursionsformel

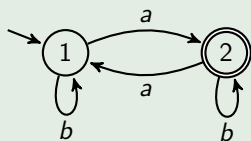
$$\gamma_{1,2}^1 = \gamma_{1,2}^0 | \gamma_{1,1}^0 (\gamma_{1,1}^0)^* \gamma_{1,2}^0$$

$r$	$(p, q)$			
	(1, 1)	(1, 2)	(2, 1)	(2, 2)
0	$\epsilon   b$	$a$	$a$	$\epsilon   b$
1	-			
2				

# Charakterisierung von REG durch reguläre Ausdrücke

## Beispiel (Fortsetzung)

DFA  $M$



Rekursionsformel

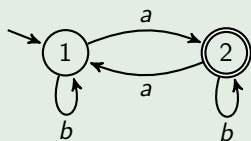
$$\begin{aligned}\gamma_{1,2}^1 &= \gamma_{1,2}^0 | \gamma_{1,1}^0 (\gamma_{1,1}^0)^* \gamma_{1,2}^0 \\ &= a | (\epsilon | b) (\epsilon | b)^* a\end{aligned}$$

$r$	$(p, q)$			
	(1, 1)	(1, 2)	(2, 1)	(2, 2)
0	$\epsilon   b$	$a$	$a$	$\epsilon   b$
1	-			
2				

# Charakterisierung von REG durch reguläre Ausdrücke

## Beispiel (Fortsetzung)

DFA  $M$



Rekursionsformel

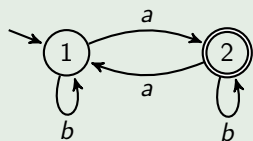
$$\begin{aligned} \gamma_{1,2}^1 &= \gamma_{1,2}^0 | \gamma_{1,1}^0 (\gamma_{1,1}^0)^* \gamma_{1,2}^0 \\ &= a | (\epsilon | b) (\epsilon | b)^* a \\ &\equiv b^* a \end{aligned}$$

$r$	$(p, q)$			
	(1, 1)	(1, 2)	(2, 1)	(2, 2)
0	$\epsilon   b$	$a$	$a$	$\epsilon   b$
1	-	$b^* a$	-	
2				

# Charakterisierung von REG durch reguläre Ausdrücke

## Beispiel (Fortsetzung)

DFA  $M$



Rekursionsformel

$$\gamma_{2,2}^1 = \gamma_{2,2}^0 | \gamma_{2,1}^0 (\gamma_{1,1}^0)^* \gamma_{1,2}^0$$

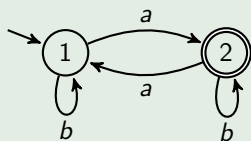
$r$	$(p, q)$			
	(1, 1)	(1, 2)	(2, 1)	(2, 2)
0	$\epsilon   b$	$a$	$a$	$\epsilon   b$
1	-	$b^* a$	-	
2				



# Charakterisierung von REG durch reguläre Ausdrücke

## Beispiel (Fortsetzung)

DFA  $M$



Rekursionsformel

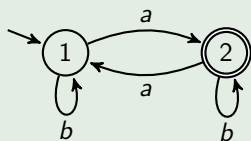
$$\begin{aligned} \gamma_{2,2}^1 &= \gamma_{2,2}^0 | \gamma_{2,1}^0 (\gamma_{1,1}^0)^* \gamma_{1,2}^0 \\ &= (\epsilon | b) | a (\epsilon | b)^* a \end{aligned}$$

$r$	$(p, q)$			
	(1, 1)	(1, 2)	(2, 1)	(2, 2)
0	$\epsilon   b$	$a$	$a$	$\epsilon   b$
1	-	$b^* a$	-	
2				

# Charakterisierung von REG durch reguläre Ausdrücke

## Beispiel (Fortsetzung)

DFA  $M$



Rekursionsformel

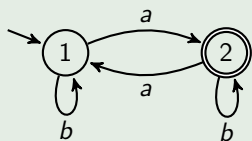
$$\begin{aligned}r_{2,2}^1 &= r_{2,2}^0 | r_{2,1}^0 (r_{1,1}^0)^* r_{1,2}^0 \\ &= (\epsilon | b) | a (\epsilon | b)^* a \\ &\equiv \epsilon | b | a b^* a\end{aligned}$$

$r$	$(p, q)$			
	(1, 1)	(1, 2)	(2, 1)	(2, 2)
0	$\epsilon   b$	$a$	$a$	$\epsilon   b$
1	-	$b^* a$	-	$\epsilon   b   a b^* a$
2	-			

# Charakterisierung von REG durch reguläre Ausdrücke

## Beispiel (Fortsetzung)

DFA  $M$



Rekursionsformeln

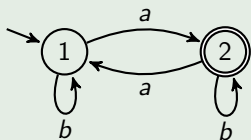
$$\gamma_{1,2}^2 = \gamma_{1,2}^1 | \gamma_{1,2}^1 (\gamma_{2,2}^1)^* \gamma_{2,2}^1$$

$r$	$(p, q)$			
	(1, 1)	(1, 2)	(2, 1)	(2, 2)
0	$\epsilon   b$	$a$	$a$	$\epsilon   b$
1	-	$b^* a$	-	$\epsilon   b   ab^* a$
2	-			

# Charakterisierung von REG durch reguläre Ausdrücke

## Beispiel (Fortsetzung)

DFA  $M$



Rekursionsformel

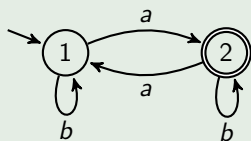
$$\begin{aligned}\gamma_{1,2}^2 &= \gamma_{1,2}^1 | \gamma_{1,2}^1 (\gamma_{2,2}^1)^* \gamma_{2,2}^1 \\ &= b^* a | b^* a (\epsilon | b | a b^* a)^* (\epsilon | b | a b^* a)\end{aligned}$$

$r$	$(p, q)$			
	(1, 1)	(1, 2)	(2, 1)	(2, 2)
0	$\epsilon   b$	$a$	$a$	$\epsilon   b$
1	-	$b^* a$	-	$\epsilon   b   a b^* a$
2	-			

# Charakterisierung von REG durch reguläre Ausdrücke

## Beispiel (Fortsetzung)

DFA  $M$



Rekursionsformel

$$\begin{aligned} \gamma_{1,2}^2 &= \gamma_{1,2}^1 | \gamma_{1,2}^1 (\gamma_{2,2}^1)^* \gamma_{2,2}^1 \\ &= b^* a | b^* a (\epsilon | b | ab^* a)^* (\epsilon | b | ab^* a) \\ &\equiv b^* a (b | ab^* a)^* \end{aligned}$$

$r$	$(p, q)$			
	(1, 1)	(1, 2)	(2, 1)	(2, 2)
0	$\epsilon   b$	$a$	$a$	$\epsilon   b$
1	-	$b^* a$	-	$\epsilon   b   ab^* a$
2	-	$b^* a (b   ab^* a)^*$	-	-

# Charakterisierungen der Klasse REG

## Korollar

Sei  $L$  eine Sprache. Dann sind folgende Aussagen äquivalent:

- $L$  ist regulär,
- es gibt einen DFA  $M$  mit  $L = L(M)$ ,
- es gibt einen NFA  $N$  mit  $L = L(N)$ ,
- es gibt einen regulären Ausdruck  $\gamma$  mit  $L = L(\gamma)$ ,
- $L$  lässt sich mit den Operationen Vereinigung, Produkt und Sternhülle aus endlichen Sprachen gewinnen,
- $L$  lässt sich mit den Operationen  $\cap$ ,  $\cup$ , Komplement, Produkt und Sternhülle aus endlichen Sprachen gewinnen.

## Ausblick

- Als nächstes wenden wir uns der Frage zu, wie sich die Anzahl der Zustände eines DFA minimieren lässt.
- Da hierbei Äquivalenzrelationen eine wichtige Rolle spielen, befassen wir uns zunächst mit Relationalstrukturen.

# Relationalstrukturen

## Definition

- Sei  $A$  eine nichtleere Menge,  $R$  ist eine  $k$ -stellige Relation auf  $A$ , wenn  $R \subseteq A^k = \underbrace{A \times \dots \times A}_{k\text{-mal}}$  ist.
- Für  $i = 1, \dots, n$  sei  $R_i$  eine  $k_i$ -stellige Relation auf  $A$ . Dann heißt  $(A; R_1, \dots, R_n)$  **Relationalstruktur**.
- Die Menge  $A$  heißt der **Individuenbereich**, die **Trägermenge** oder die **Grundmenge** der Relationalstruktur.

## Bemerkung

- Wir werden hier hauptsächlich den Fall  $n = 1$ ,  $k_1 = 2$ , also  $(A, R)$  mit  $R \subseteq A \times A$  betrachten.
- Man nennt dann  $R$  eine **(binäre) Relation** auf  $A$ .
- Oft wird für  $(a, b) \in R$  auch die **Infix-Schreibweise**  $aRb$  benutzt.

# Relationalstrukturen

## Beispiel

- $(F, M)$  mit  $F = \{f \mid f \text{ ist Fluss in Europa}\}$  und
$$M = \{(f, g) \in F \times F \mid f \text{ mündet in } g\},$$
- $(U, B)$  mit  $U = \{x \mid x \text{ ist Berliner}\}$  und
$$B = \{(x, y) \in U \times U \mid x \text{ ist Bruder von } y\},$$
- $(\mathcal{P}(M), \subseteq)$ , wobei  $M$  eine beliebige Menge und  $\subseteq$  die Inklusionsrelation auf den Teilmengen von  $M$  ist,
- $(A, Id_A)$  mit  $Id_A = \{(x, x) \mid x \in A\}$  (die **Identität auf  $A$** ),
- $(\mathbb{R}, \leq)$ ,
- $(\mathbb{Z}, \mid)$ , wobei  $\mid$  die "teilt"-Relation bezeichnet,
- $(Fml, \Rightarrow)$  mit  $Fml = \{F \mid F \text{ ist aussagenlogische Formel}\}$  und
$$\Rightarrow = \{(F, G) \in Fml \times Fml \mid G \text{ ist log. Folgerung von } F\}.$$



# Mengentheoretische Operationen auf Relationen

- Da Relationen Mengen sind, können wir den **Durchschnitt**, die **Vereinigung**, die **Differenz** und das **Komplement** von Relationen bilden:

$$R \cap S = \{(x, y) \in A \times A \mid xRy \wedge xSy\},$$

$$R \cup S = \{(x, y) \in A \times A \mid xRy \vee xSy\},$$

$$R - S = \{(x, y) \in A \times A \mid xRy \wedge \neg xSy\},$$

$$\bar{R} = (A \times A) - R.$$

- Sei  $\mathcal{M} \subseteq \mathcal{P}(A \times A)$  eine beliebige Menge von Relationen auf  $A$ . Dann sind der **Schnitt über  $\mathcal{M}$**  und die **Vereinigung über  $\mathcal{M}$**  folgende Relationen:

$$\bigcap \mathcal{M} = \{(x, y) \mid \forall R \in \mathcal{M} : xRy\},$$

$$\bigcup \mathcal{M} = \{(x, y) \mid \exists R \in \mathcal{M} : xRy\}.$$

- Weiterhin ist die **Inklusion  $R \subseteq S$**  auf Relationen definiert. Es gilt

$$R \subseteq S \Leftrightarrow \forall x, y : xRy \rightarrow xSy.$$

# Weitere Operationen auf Relationen

## Definition

- Die **transponierte (konverse) Relation** zu  $R$  ist

$$R^T = \{(y, x) \mid xRy\}.$$

- $R^T$  wird oft auch mit  $R^{-1}$  bezeichnet.
- Zum Beispiel ist  $(\mathbb{R}, \leq^T) = (\mathbb{R}, \geq)$ .
- Das **Produkt** (oder die **Komposition**) zweier Relationen  $R$  und  $S$  ist

$$R \circ S = \{(x, z) \in A \times A \mid \exists y \in A : xRy \wedge ySz\}.$$

## Beispiel

Ist  $B$  die Relation "ist Bruder von",  $V$  "ist Vater von",  $M$  "ist Mutter von" und  $E = V \cup M$  "ist Elternteil von", so ist  $B \circ E$  die Onkel-Relation.  $\triangleleft$

# Das Relationenprodukt

## Notation

- Für  $R \circ S$  wird auch  $R ; S$ ,  $R \cdot S$  oder einfach  $RS$  geschrieben.
- Für  $\underbrace{R \circ \dots \circ R}_{n\text{-mal}}$  schreiben wir auch  $R^n$ . Dabei ist  $R^0 = Id$ .

## Vorsicht!

Das Relationenprodukt  $R^n$  sollte nicht mit dem kartesischen Produkt

$$\underbrace{R \times \dots \times R}_{n\text{-mal}}$$

verwechselt werden.

## Vereinbarung

Wir vereinbaren, dass  $R^n$  das  $n$ -fache Relationenprodukt bezeichnen soll, falls  $R$  eine Relation ist.

# Eigenschaften von Relationen

## Definition

Sei  $R$  eine Relation auf  $A$ . Dann heißt  $R$

**reflexiv**, falls  $\forall x \in A : xRx$  (also  $Id_A \subseteq R$ )

**irreflexiv**, falls  $\forall x \in A : \neg xRx$  (also  $Id_A \subseteq \bar{R}$ )

**symmetrisch**, falls  $\forall x, y \in A : xRy \Rightarrow yRx$  (also  $R \subseteq R^T$ )

**asymmetrisch**, falls  $\forall x, y \in A : xRy \Rightarrow \neg yRx$  (also  $R \subseteq \overline{R^T}$ )

**antisymmetrisch**, falls  $\forall x, y \in A : xRy \wedge yRx \Rightarrow x = y$  (also  $R \cap R^T \subseteq Id$ )

**konnex**, falls  $\forall x, y \in A : xRy \vee yRx$  (also  $A \times A \subseteq R \cup R^T$ )

**semikonnex**, falls  $\forall x, y \in A : x \neq y \Rightarrow xRy \vee yRx$  (also  $\bar{Id} \subseteq R \cup R^T$ )

**transitiv**, falls  $\forall x, y, z \in A : xRy \wedge yRz \Rightarrow xRz$  (also  $R^2 \subseteq R$ )

gilt.

# Eigenschaften von Relationen

## Beispiel

- Die Relation "ist Schwester von" ist zwar in einer reinen Damengesellschaft symmetrisch, i.a. jedoch weder symmetrisch noch asymmetrisch noch antisymmetrisch.
- $(\mathbb{R}, <)$  ist irreflexiv, asymmetrisch, transitiv und semikonnex.
- $(\mathbb{R}, \leq)$  und  $(\mathcal{P}(M), \subseteq)$  sind reflexiv, antisymmetrisch und transitiv.
- $(\mathbb{R}, \leq)$  ist auch konnex.
- $(\mathcal{P}(M), \subseteq)$  ist zwar im Fall  $\|M\| \leq 1$  konnex, aber im Fall  $\|M\| \geq 2$  weder semikonnex noch konnex.

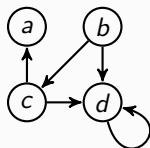


# Darstellung von endlichen Relationen

## Graphische Darstellung

$$A = \{a, b, c, d\}$$

$$R = \{(b, c), (b, d), (c, a), (c, d), (d, d)\}$$



- Eine Relation  $R$  auf einer endlichen Menge  $A$  kann durch einen **gerichteten Graphen** (kurz **Digraphen**)  $G = (A, R)$  mit **Knotenmenge**  $A$  und **Kantenmenge**  $R$  veranschaulicht werden.
- Hierzu stellen wir jedes Element  $x \in A$  als einen Knoten dar und verbinden jedes Knotenpaar  $(x, y) \in R$  durch eine gerichtete Kante (Pfeil).
- Zwei durch eine Kante verbundene Knoten heißen **adjazent** oder **benachbart**.

# Darstellung von endlichen Relationen

## Definition

Sei  $R$  eine binäre Relation auf  $A$ .

- Der **Nachbereich** von  $x$  ist

$$R(x) = \{y \in A \mid xRy\}.$$

- Der **Ausgangsgrad** eines Knotens  $x$  ist

$$\deg^+(x) = \|R(x)\|.$$

- Der **Eingangsgrad** von  $x$  ist

$$\deg^-(x) = \|\{y \in A \mid yRx\}\| = \|R^{-1}(x)\|.$$

- Ist  $R$  symmetrisch, so können wir die Pfeilspitzen auch weglassen.
- In diesem Fall ist  $\deg(x) = \deg^-(x) = \deg^+(x)$  der **Grad** von  $x$  in  $G$ .
- $G$  ist **schleifenfrei**, falls  $R$  irreflexiv ist.
- Ist  $R$  irreflexiv und symmetrisch, so nennen wir  $G$  einen (**ungerichteten**) **Graphen**.

# Darstellung von endlichen Relationen

## Matrixdarstellung (Adjazenzmatrix)

Eine Relation  $R$  auf  $A = \{a_1, \dots, a_n\}$  lässt sich auch durch die boolesche  $(n \times n)$ -Matrix  $M_R = (m_{ij})$  darstellen mit

$$m_{ij} = \begin{cases} 1, & a_i R a_j, \\ 0, & \text{sonst.} \end{cases}$$

## Beispiel

Die Relation  $R = \{(b, c), (b, d), (c, a), (c, d), (d, d)\}$  auf  $A = \{a, b, c, d\}$  hat beispielsweise die Matrixdarstellung

$$M_R = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$



# Darstellung von endlichen Relationen

## Tabellendarstellung (Adjazenzliste)

$R$  lässt sich auch durch eine Tabelle darzustellen, die jedem Element  $x \in A$  seinen Nachbereich  $R(x)$  in Form einer Liste zuordnet.

### Beispiel

Die Relation  $R = \{(b, c), (b, d), (c, a), (c, d), (d, d)\}$  auf  $A = \{a, b, c, d\}$  hat beispielsweise die Tabellendarstellung

$x$	$R(x)$
$a$	-
$b$	$c, d$
$c$	$a, d$
$d$	$d$

# Berechnung des Relationenprodukts

## Berechnung von $R \circ S$

- Sind  $M_R = (r_{ij})$  und  $M_S = (s_{ij})$  boolesche  $n \times n$ -Matrizen für  $R$  und  $S$ , so erhalten wir für  $T = R \circ S$  die Matrix  $M_T = (t_{ij})$  mit

$$t_{ij} = \bigvee_{k=1, \dots, n} (r_{ik} \wedge s_{kj}).$$

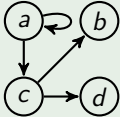
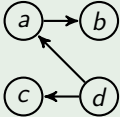
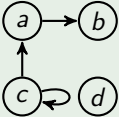
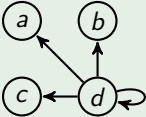
- Der Nachbereich  $T(x)$  von  $x$  bzgl. der Relation  $T = R \circ S$  berechnet sich zu

$$T(x) = \bigcup_{y \in R(x)} S(y).$$

# Das Relationsprodukt

## Beispiel

Betrachte die Relationen  $R = \{(a, a), (a, c), (c, b), (c, d)\}$  und  $S = \{(a, b), (d, a), (d, c)\}$  auf der Menge  $A = \{a, b, c, d\}$ .

Relation	$R$	$S$	$R \circ S$	$S \circ R$
Digraph				
Adjazenzmatrix	$\begin{matrix} 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{matrix}$	$\begin{matrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{matrix}$	$\begin{matrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{matrix}$	$\begin{matrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{matrix}$
Adjazenzliste	$\begin{matrix} a : a, c \\ b : - \\ c : b, d \\ d : - \end{matrix}$	$\begin{matrix} a : b \\ b : - \\ c : - \\ d : a, c \end{matrix}$	$\begin{matrix} a : b \\ b : - \\ c : a, c \\ d : - \end{matrix}$	$\begin{matrix} a : - \\ b : - \\ c : - \\ d : a, b, c, d \end{matrix}$

# Das Relationsprodukt

## Beobachtung

Das Relationsprodukt ist nicht kommutativ, d.h. i.a. gilt nicht  $R \circ S = S \circ R$ .

## Relationenalgebra

Als nächstes zeigen wir, dass die Menge  $\mathcal{R} = \mathcal{P}(A \times A)$  aller binären Relationen auf  $A$  mit dem Relationsprodukt  $\circ$  als binärer Operation und der Relation  $Id_A$  als neutralem Element eine Halbgruppe (oder **Monoid**) bildet.

## Satz

Seien  $Q, R, S$  Relationen auf  $A$ . Dann gilt

- 1  $(Q \circ R) \circ S = Q \circ (R \circ S)$ , d.h.  $\circ$  ist assoziativ,
- 2  $Id \circ R = R \circ Id = R$ , d.h.  $Id$  ist neutrales Element.

# Relationenalgebra

## Satz

Seien  $Q, R, S$  Relationen auf  $A$ . Dann gilt

- 1  $(Q \circ R) \circ S = Q \circ (R \circ S)$ , d.h.  $\circ$  ist assoziativ,
- 2  $Id \circ R = R \circ Id = R$ , d.h.  $Id$  ist neutrales Element.

## Beweis.

- 1 
$$\begin{aligned}x (Q \circ R) \circ S y &\Leftrightarrow \exists u : x (Q \circ R) u \wedge u S y \\&\Leftrightarrow \exists u : (\exists v : x Q v R u) \wedge u S y \\&\Leftrightarrow \exists u, v : x Q v R u S y \\&\Leftrightarrow \exists v : x Q v \wedge (\exists u : v R u \wedge u S y) \\&\Leftrightarrow \exists v : x Q v (R \circ S) y \\&\Leftrightarrow x Q \circ (R \circ S) y\end{aligned}$$
- 2 Wegen  $x Id \circ R y \Leftrightarrow \exists z : x = z \wedge z R y \Leftrightarrow x R y$  folgt  $Id \circ R = R$ .  
Die Gleichheit  $R \circ Id = R$  folgt analog. □

# Hüllenoperatoren

## Bemerkung

- Es ist leicht zu sehen, dass der Schnitt von transitiven Relationen ebenfalls transitiv ist.
- Die **transitive Hülle** von  $R$  ist

$$R^+ = \bigcap \{S \subseteq A \times A \mid S \text{ ist transitiv und } R \subseteq S\}.$$

- $R^+$  ist also eine transitive Relation, die  $R$  enthält.
- Da  $R^+$  zudem in jeder Relation mit diesen Eigenschaften enthalten ist, ist  $R^+$  die kleinste transitive Relation, die  $R$  enthält.
- Da auch die Reflexivität und die Symmetrie bei der Schnittbildung erhalten bleiben, lassen sich nach demselben Muster weitere Hüllenoperatoren definieren.

# Weitere Hüllenoperatoren

## Bemerkung

- Die **reflexive Hülle** von  $R$  ist

$$h_{\text{refl}}(R) = \bigcap \{S \subseteq A \times A \mid S \text{ ist reflexiv und } R \subseteq S\}.$$

- Die **symmetrische Hülle** von  $R$  ist

$$h_{\text{sym}}(R) = \bigcap \{S \subseteq A \times A \mid S \text{ ist symmetrisch und } R \subseteq S\}.$$

- Die **reflexiv-transitive Hülle** von  $R$  ist

$$R^* = \bigcap \{S \subseteq A \times A \mid S \text{ ist reflexiv, transitiv und } R \subseteq S\}.$$

# Transitive und reflexive Hülle

## Satz

$$h_{\text{refl}}(R) = R \cup \text{Id}_A, \quad h_{\text{sym}}(R) = R \cup R^T, \quad R^+ = \bigcup_{n \geq 1} R^n, \quad R^* = \bigcup_{n \geq 0} R^n.$$

## Beweis

Siehe Übungen. □

## Bemerkung

- Ein Paar  $(a, b)$  ist also genau dann in der reflexiv-transitiven Hülle  $R^*$  von  $R$  enthalten, wenn es ein  $n \geq 0$  gibt mit  $aR^n b$ .
- Dies bedeutet, dass es Elemente  $x_0, \dots, x_n \in A$  gibt mit
$$x_0 = a, x_n = b \text{ und } x_0 R x_1 R x_2 \cdots R x_{n-1} R x_n.$$
- $x_0, \dots, x_n$  heißt **Weg** der Länge  $n$  von  $a$  nach  $b$ .



# Überblick über Relationalstrukturen

## Äquivalenz- und Ordnungsrelationen

	refl.	sym.	trans.	antisym.	asym.	konnex	semikon.
Äquivalenzrelation	✓	✓	✓				
(Halb-)Ordnung	✓		✓	✓			
Striktordnung			✓			✓	
lineare Ordnung			✓	✓		✓	
lin. Striktord.			✓			✓	✓
Quasiordnung	✓		✓				

## Bemerkung

In der Tabelle sind nur die definierenden Eigenschaften durch ein "✓" gekennzeichnet. Das schließt nicht aus, dass noch weitere Eigenschaften vorliegen.

# Äquivalenzrelationen

## Beispiel

- Auf der Menge aller Geraden im  $\mathbb{R}^2$  die Parallelität.
- Auf der Menge aller Menschen "im gleichen Jahr geboren wie".
- Auf  $\mathbb{Z}$  die Relation "gleicher Rest bei Division durch  $m$ ".
- Auf der Menge aller aussagenlogischen Formeln die semantische Äquivalenz.



# Äquivalenzrelationen

## Definition

- Ist  $E$  eine Äquivalenzrelation, so nennt man den Nachbereich  $E(x)$  die **von  $x$  repräsentierte Äquivalenzklasse** und bezeichnet ihn auch mit  $[x]_E$  oder einfach mit  $[x]$ :

$$[x]_E = [x] = \{x' \mid xEx'\}.$$

- Die durch  $E$  induzierte Zerlegung (Partition)  $\{[x]_E \mid x \in A\}$  von  $A$  wird **Quotienten- oder Faktormenge** genannt und mit  $A/E$  bezeichnet:

$$A/E = \{[x]_E \mid x \in A\}.$$

- Die Anzahl  $\|A/E\|$  der Äquivalenzklassen von  $E$  wird auch als der **Index** von  $E$  bezeichnet.
- Eine Menge  $S \subseteq A$  heißt **Repräsentantensystem**, falls sie genau ein Element aus jeder Äquivalenzklasse enthält.

# Äquivalenzrelationen

## Beispiel

Für die weiter oben betrachteten Äquivalenzrelationen erhalten wir folgende Klasseneinteilungen:

- Für die Parallelität auf der Menge aller Geraden im  $\mathbb{R}^2$ : alle Geraden mit derselben Richtung (oder Steigung) bilden jeweils eine Äquivalenzklasse.
- Ein Repräsentantensystem wird beispielsweise durch die Menge aller Ursprungsgeraden gebildet.
- Für die Relation "im gleichen Jahr geboren wie" auf der Menge aller Menschen: jeder Jahrgang bildet eine Äquivalenzklasse.
- Für die Relation "gleicher Rest bei Division durch  $m$ " auf  $\mathbb{Z}$ : jede der  $m$  Restklassen  $[0], [1], \dots, [m-1]$  mit

$$[r] = \{a \in \mathbb{Z} \mid a \bmod m = r\}$$

bildet eine Äquivalenzklasse.

- Repräsentantensystem:  $\{0, 1, \dots, m-1\}$ .

# Partition einer Menge

## Definition

Eine Familie  $\{M_i \mid i \in I\}$  von nichtleeren Teilmengen  $M_i \subseteq A$  heißt **Partition** der Menge  $A$ , falls gilt:

- die Mengen  $M_i$  **überdecken**  $A$ , d.h.  $A = \bigcup_{i \in I} M_i$  und
- die Mengen  $M_i$  sind **paarweise disjunkt**, d.h. für je zwei verschiedene Mengen  $M_i \neq M_j$  gilt  $M_i \cap M_j = \emptyset$ .

## Satz

Sei  $E$  eine Relation auf  $A$ . Dann sind folgende Aussagen äquivalent:

- 1  $E$  ist eine Äquivalenzrelation auf  $A$ .
- 2 Für alle  $x, y \in A$  gilt  $xEy \Leftrightarrow E(x) = E(y)$ ,
- 3  $E$  ist reflexiv und  $\{E(x) \mid x \in A\}$  ist eine Partition von  $A$ .

# Charakterisierung von Äquivalenzrelationen

## Satz

Sei  $E$  eine Relation auf  $A$ . Dann sind folgende Aussagen äquivalent:

- 1  $E$  ist eine Äquivalenzrelation auf  $A$ .
- 2 Für alle  $x, y \in A$  gilt  $xEy \Leftrightarrow E(x) = E(y)$ ,
- 3  $E$  ist reflexiv und  $\{E(x) \mid x \in A\}$  ist eine Partition von  $A$ .

## Beweis.

1 impliziert 2: Sei  $E$  eine Äquivalenzrelation auf  $A$ .

Da  $E$  transitiv ist, impliziert  $xEy$  die Inklusion  $E(y) \subseteq E(x)$ :

$$z \in E(y) \Rightarrow yEz \Rightarrow xEz \Rightarrow z \in E(x).$$

Da  $E$  symmetrisch ist, folgt aus  $xEy$  aber auch  $E(x) \subseteq E(y)$ .

Umgekehrt folgt aus  $E(x) = E(y)$  wegen der Reflexivität von  $E$ , dass  $x \in E(x) = E(y)$  enthalten ist, und somit  $xEy$ .

# Charakterisierung von Äquivalenzrelationen

## Satz

Sei  $E$  eine Relation auf  $A$ . Dann sind folgende Aussagen äquivalent:

- 1  $E$  ist eine Äquivalenzrelation auf  $A$ .
- 2 Für alle  $x, y \in A$  gilt  $xEy \Leftrightarrow E(x) = E(y)$ ,
- 3  $E$  ist reflexiv und  $\{E(x) \mid x \in A\}$  ist eine Partition von  $A$ .

## Beweis.

2 impliziert 3: Falls  $E$  die Bedingung  $xEy \Leftrightarrow E(x) = E(y)$  erfüllt, so folgt sofort  $xEx$  (wegen  $E(x) = E(x)$ ) und folglich überdecken die Nachbereiche  $E(x)$  (wegen  $x \in E(x)$ ) die Menge  $A$ .

Ist  $E(x) \cap E(y) \neq \emptyset$  und  $z$  ein Element in  $E(x) \cap E(y)$ , so gilt  $xEz$  und  $yEz$  und daher folgt  $E(x) = E(z) = E(y)$ .

Da also je zwei Nachbereiche  $E(x)$  und  $E(y)$  entweder gleich oder disjunkt sind, bildet  $\{E(x) \mid x \in A\}$  sogar eine Partition von  $A$ .

# Charakterisierung von Äquivalenzrelationen

## Satz

Sei  $E$  eine Relation auf  $A$ . Dann sind folgende Aussagen äquivalent:

- 1  $E$  ist eine Äquivalenzrelation auf  $A$ .
- 2 Für alle  $x, y \in A$  gilt  $xEy \Leftrightarrow E(x) = E(y)$ ,
- 3  $E$  ist reflexiv und  $\{E(x) \mid x \in A\}$  ist eine Partition von  $A$ .

## Beweis.

3 impliziert 1: Wird schließlich  $A$  von den Mengen  $E(x)$  partitioniert, wobei  $x \in E(x)$  für alle  $x \in A$  gilt, so folgt

$$xEy \Leftrightarrow y \in E(x) \cap E(y) \Leftrightarrow E(x) = E(y).$$

Daher übertragen sich die Eigenschaften Reflexivität, Symmetrie und Transitivität unmittelbar von der Gleichheitsrelation auf  $E$ . □



# Verfeinerung und Vergrößerung von Äquivalenzrelationen

## Bemerkungen

- Die kleinste Äquivalenzrelation auf  $A$  ist die Identität  $Id_A$ , die größte ist die Allrelation  $A \times A$ .
- Die Äquivalenzklassen der Identität enthalten jeweils nur ein Element, d.h.  $A/Id_A = \{\{x\} \mid x \in A\}$ .
- Die Allrelation erzeugt nur eine Äquivalenzklasse, nämlich  $A$ , d.h.  $A/(A \times A) = \{A\}$ .
- Für zwei Äquivalenzrelationen  $E \subseteq E'$  sind auch die Äquivalenzklassen  $[x]_E$  von  $E$  in den Klassen  $[x]_{E'}$  von  $E'$  enthalten.
- Folglich ist jede Äquivalenzklasse von  $E'$  die Vereinigung von (evtl. mehreren) Äquivalenzklassen von  $E$ .
- Im Fall  $E \subseteq E'$  sagt man auch,  $E$  bewirkt eine **feinere** Partitionierung als  $E'$ .
- Demnach ist die Identität die **feinste** und die Allrelation die **größte** Äquivalenzrelation.

# Schwache Zusammenhangskomponenten eines Digraphen

## Bemerkung

- Der Schnitt über eine Menge von Äquivalenzrelationen auf  $A$  ist wieder eine Äquivalenzrelation auf  $A$ .
- Daher können wir für eine beliebige Relation  $R$  auf  $A$  die kleinste  $R$  umfassende Äquivalenzrelation auf  $A$  definieren:

$$h_{\text{äq}}(R) = \bigcap \{E \subseteq A \times A \mid E \text{ ist eine Äquivalenzrelation mit } R \subseteq E\}.$$

- In der Graphentheorie werden die durch die Äquivalenzklassen von  $h_{\text{äq}}(R)$  induzierten Teilgraphen auch die **schwachen Zusammenhangskomponenten** des Digraphen  $(A, R)$  genannt.

# Ordnungsrelationen

## Definition

$(A, R)$  heißt **Ordnung** (auch **Halbordnung** oder **partielle Ordnung**), wenn  $R$  eine reflexive, antisymmetrische und transitive Relation auf  $A$  ist.

## Beispiel

- $(\mathcal{P}(M); \subseteq)$ ,  $(\mathbb{Z}, \leq)$ ,  $(\mathbb{N}, |)$ .
- Auf der Menge  $\mathcal{A}(M)$  aller Äquivalenzrelationen auf  $M$  die Relation "feiner als".
- Ist  $R$  eine Ordnung auf  $A$  und  $B \subseteq A$ , so heißt die Ordnung  $R_B = R \cap (B \times B)$  die **Einschränkung** (oder **Restriktion**) von  $R$  auf  $B$ .
- Beispielsweise ist  $(\mathcal{A}(M); \subseteq)$  die Einschränkung von  $(\mathcal{P}(M \times M); \subseteq)$  auf  $\mathcal{A}(M)$ .



# Darstellung von Ordnungen durch ein Hasse-Diagramm

- Sei  $\leq$  eine Ordnung auf  $A$  und sei  $<$  die Relation  $\leq \cap \overline{Id}_A$ .
- Um  $\leq$  in einem **Hasse-Diagramm** darzustellen, wird nur der Graph der **Nachbarrelation**

$$\triangleleft = < \setminus <^2, \text{ d.h. } x \triangleleft y \Leftrightarrow x < y \wedge \neg \exists z : x < z < y$$

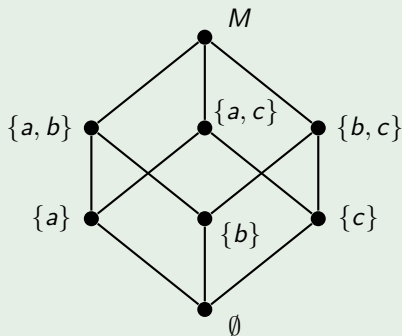
gezeichnet.

- Für  $x \triangleleft y$  sagt man auch,  $y$  ist **oberer Nachbar** von  $x$ .
- Weiterhin wird im Fall  $x \triangleleft y$  der Knoten  $y$  oberhalb vom Knoten  $x$  gezeichnet, so dass auf Pfeilspitzen verzichtet werden kann.

# Das Hasse-Diagramm für $(\mathcal{P}(M); \subseteq)$

## Beispiel

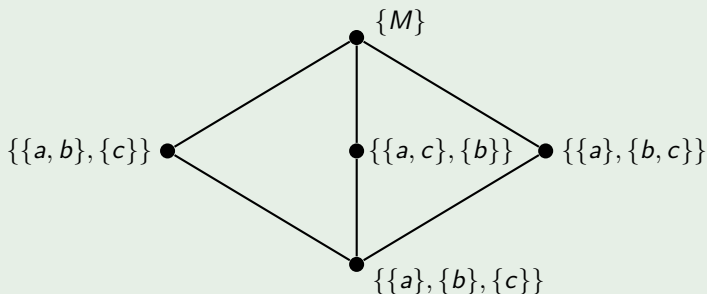
Die Inklusion  $\subseteq$  auf  $\mathcal{P}(M)$  mit  $M = \{a, b, c\}$  lässt sich durch folgendes Hasse-Diagramm darstellen:



# Das Hasse-Diagramm für $(\mathcal{A}(M); \subseteq)$

## Beispiel

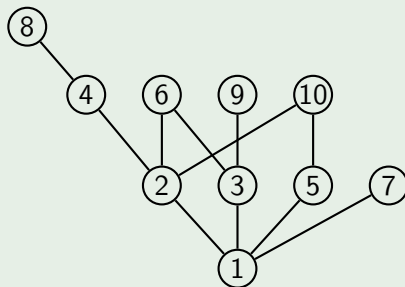
Die "feiner als" Relation auf der Menge aller Partitionen von  $M = \{a, b, c\}$  ist durch folgendes Hasse-Diagramm darstellbar:



# Das Hasse-Diagramm der "teilt"-Relation

## Beispiel

Die "teilt"-Relation auf  $\{1, 2, \dots, 10\}$  ist durch folgendes Hasse-Diagramm darstellbar:



# Maximale, minimale, größte und kleinste Elemente

## Definition

Sei  $\leq$  eine Ordnung auf  $A$  und sei  $b$  ein Element in einer Teilmenge  $B \subseteq A$ .

- $b$  heißt **kleinstes Element** oder **Minimum** von  $B$ , falls gilt:

$$\forall b' \in B : b \leq b'.$$

- $b$  heißt **größtes Element** oder **Maximum** von  $B$ , falls gilt:

$$\forall b' \in B : b' \leq b.$$

- $b$  heißt **minimal** in  $B$ , falls es in  $B$  kein kleineres Element gibt:

$$\forall b' \in B : b' \leq b \Rightarrow b' = b.$$

- $b$  heißt **maximal** in  $B$ , falls es in  $B$  kein größeres Element gibt:

$$\forall b' \in B : b \leq b' \Rightarrow b = b'.$$



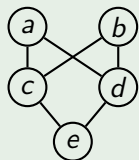
# Maximale, minimale, größte und kleinste Elemente

## Bemerkung

Wegen der Antisymmetrie kann es in  $B$  höchstens ein kleinstes und höchstens ein größtes Element geben.

## Beispiel

Betrachte folgende Ordnung.



$B$	minimal in $B$	maximal in $B$	Minimum von $B$	Maximum von $B$
$\{a, b\}$	$a, b$	$a, b$	-	-
$\{c, d\}$	$c, d$	$c, d$	-	-
$\{a, b, c\}$	$c$	$a, b$	$c$	-
$\{a, b, c, e\}$	$e$	$a, b$	$e$	-
$\{a, c, d, e\}$	$e$	$a$	$e$	$a$

# Obere und untere Schranken

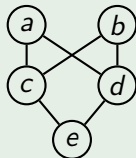
## Definition

Sei  $\leq$  eine Ordnung auf  $A$  und sei  $B \subseteq A$ .

- Jedes Element  $u \in A$  mit  $u \leq b$  für alle  $b \in B$  heißt **untere Schranke** von  $B$ .
- Jedes Element  $o \in A$  mit  $b \leq o$  für alle  $b \in B$  heißt **obere Schranke** von  $B$ .
- $B$  heißt **nach oben beschränkt**, wenn  $B$  eine obere Schranke hat.
- $B$  heißt **nach unten beschränkt**, wenn  $B$  eine untere Schranke hat.
- $B$  heißt **beschränkt**, wenn  $B$  nach oben und nach unten beschränkt ist.

# Obere und untere Schranken

## Beispiel (Fortsetzung)



$B$	minimal	maximal	min	max	untere Schranken	obere Schranken
$\{a, b\}$	$a, b$	$a, b$	-	-	$c, d, e$	-
$\{c, d\}$	$c, d$	$c, d$	-	-	$e$	$a, b$
$\{a, b, c\}$	$c$	$a, b$	$c$	-	$c, e$	-
$\{a, b, c, e\}$	$e$	$a, b$	$e$	-	$e$	-
$\{a, c, d, e\}$	$e$	$a$	$e$	$a$	$e$	$a$

# Infima und Suprema

## Definition

Sei  $\leq$  eine Ordnung auf  $A$  und sei  $B \subseteq A$ .

- Besitzt  $B$  eine größte untere Schranke  $i$ , d.h. besitzt die Menge  $U$  aller unteren Schranken von  $B$  ein größtes Element  $i$ , so heißt  $i$  das **Infimum von  $B$**  ( $i = \inf B$ ):

$$(\forall b \in B : b \geq i) \wedge [\forall u \in A : (\forall b \in B : b \geq u) \Rightarrow u \leq i].$$

- Besitzt  $B$  eine kleinste obere Schranke  $s$ , d.h. besitzt die Menge  $O$  aller oberen Schranken von  $B$  ein kleinstes Element  $s$ , so heißt  $s$  das **Supremum von  $B$**  ( $s = \sup B$ ):

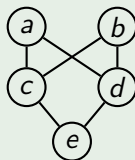
$$(\forall b \in B : b \leq s) \wedge [\forall o \in A : (\forall b \in B : b \leq o) \Rightarrow s \leq o]$$

## Bemerkung

$B$  kann nicht mehr als ein Supremum und ein Infimum haben.

# Infima und Suprema

## Beispiel (Schluss)



$B$	minimal	maximal	min	max	untere Schranken	obere Schranken	inf	sup
$\{a, b\}$	$a, b$	$a, b$	-	-	$c, d, e$	-	-	-
$\{c, d\}$	$c, d$	$c, d$	-	-	$e$	$a, b$	$e$	-
$\{a, b, c\}$	$c$	$a, b$	$c$	-	$c, e$	-	$c$	-
$\{a, b, c, e\}$	$e$	$a, b$	$e$	-	$e$	-	$e$	-
$\{a, c, d, e\}$	$e$	$a$	$e$	$a$	$e$	$a$	$e$	$a$

# Infima und Suprema

## Bemerkung

- Auch in linearen Ordnungen muss nicht jede Teilmenge ein Supremum oder Infimum besitzen.
- So hat in der linear geordneten Menge  $(\mathbb{Q}, \leq)$  die Teilmenge

$$B = \{x \in \mathbb{Q} \mid x^2 \leq 2\} = \{x \in \mathbb{Q} \mid x^2 < 2\}$$

weder ein Supremum noch ein Infimum.

- Dagegen hat in einer linearen Ordnung jede **endliche** Teilmenge ein kleinstes und ein größtes Element und somit erst recht ein Supremum und ein Infimum.

# Abbildungen

## Definition

Sei  $R$  eine binäre Relation auf einer Menge  $M$ .

- $R$  heißt **rechtseindeutig**, falls gilt:

$$\forall x, y, z \in M : xRy \wedge xRz \Rightarrow y = z.$$

- $R$  heißt **linkseindeutig**, falls gilt:

$$\forall x, y, z \in M : xRz \wedge yRz \Rightarrow x = y.$$

- Der **Nachbereich**  $N(R)$  und der **Vorbereich**  $V(R)$  von  $R$  sind

$$N(R) = \bigcup_{x \in M} R(x) \quad \text{und} \quad V(R) = \bigcup_{x \in M} R^T(x).$$

# Abbildungen

Abbildungen ordnen jedem Element ihres Definitionsbereichs genau ein Element zu.

## Definition

Eine rechtseindeutige Relation  $R$  mit  $V(R) = A$  und  $N(R) \subseteq B$  heißt **Abbildung** oder **Funktion von A nach B** (kurz  $R : A \rightarrow B$ ).

## Bemerkung

- Wie üblich werden wir Abbildungen meist mit kleinen Buchstaben  $f, g, h, \dots$  bezeichnen und für  $(x, y) \in f$  nicht  $xfy$  sondern  $f(x) = y$  oder  $f : x \mapsto y$  schreiben.
- Ist  $f : A \rightarrow B$  eine Abbildung, so wird der Vorbereich  $V(f) = A$  der **Definitionsbereich** und die Menge  $B$  der **Wertebereich** oder **Wertevorrat** von  $f$  genannt.
- Der Nachbereich  $N(f)$  wird als **Bild** von  $f$  bezeichnet.



# Abbildungen

## Definition

- Im Fall  $N(f) = B$  heißt  $f$  **surjektiv**.
- Ist  $f$  linkseindeutig, so heißt  $f$  **injektiv**.
- In diesem Fall impliziert  $f(x) = f(y)$  die Gleichheit  $x = y$ .
- Eine injektive und surjektive Abbildung heißt **bijektiv**.
- Für eine injektive Abbildung  $f : A \rightarrow B$  ist auch  $f^{-1}$  eine Abbildung, die als die zu  $f$  **inverse Abbildung** bezeichnet wird.

## Bemerkung

Man beachte, dass der Definitionsbereich  $V(f^{-1}) = N(f)$  von  $f^{-1}$  nur dann gleich  $B$  ist, wenn  $f$  auch surjektiv, also eine Bijektion ist.

# Homomorphismen

## Definition

Seien  $(A_1, R_1)$  und  $(A_2, R_2)$  Relationalstrukturen.

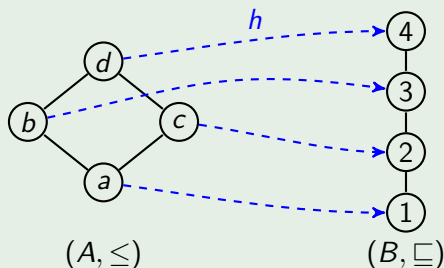
- Eine Abbildung  $h : A_1 \rightarrow A_2$  heißt **Homomorphismus**, falls für alle  $a, b \in A_1$  gilt:

$$aR_1b \Rightarrow h(a)R_2h(b).$$

- Sind  $(A_1, R_1)$  und  $(A_2, R_2)$  Ordnungen, so spricht man auch von **Ordnungshomomorphismen** oder einfach von **monotonen** Abbildungen.
- Injektive Ordnungshomomorphismen werden auch **streng monotone** Abbildungen genannt.

# Homomorphismen

## Beispiel



- Die Abbildung  $h : A \rightarrow B$  ist ein bijektiver Ordnungshomomorphismus.
- Die Umkehrabbildung  $h^{-1}$  ist jedoch kein Homomorphismus, da  $h^{-1}$  nicht monoton ist.
- Es gilt nämlich  $2 \subseteq 3$ , aber  $h^{-1}(2) = b \not\subseteq c = h^{-1}(3)$ .

# Isomorphismen

## Definition

- Seien  $(A_1, R_1)$  und  $(A_2, R_2)$  Relationalstrukturen.
- Ein bijektiver Homomorphismus  $h : A_1 \rightarrow A_2$ , bei dem auch  $h^{-1}$  ein Homomorphismus ist, d.h. es gilt

$$\forall a, b \in A_1 : aR_1b \Leftrightarrow h(a)R_2h(b).$$

heißt **Isomorphismus**.

- In diesem Fall heißen die Strukturen  $(A_1, R_1)$  und  $(A_2, R_2)$  **isomorph** (kurz:  $(A_1, R_1) \cong (A_2, R_2)$ ).

Sind  $(A_1, R_1)$  und  $(A_2, R_2)$  isomorph, so bedeutet dies, dass sich die eine Struktur aus der anderen durch eine bloße Umbenennung der Elemente gewinnen lässt.

# Isomorphismen

## Beispiel

- Die Bijektion  $h : x \mapsto e^x$  ist ein Ordnungsisomorphismus zwischen  $(\mathbb{R}, \leq)$  und  $(\mathbb{R}^+, \leq)$ .
- Für  $n \in \mathbb{N}$  sei

$$T_n = \{k \in \mathbb{N} \mid k \text{ teilt } n\}$$

und

$$P_n = \{k \in \mathbb{N} \mid k \text{ ist Primteiler von } n\}.$$

Dann ist die Abbildung

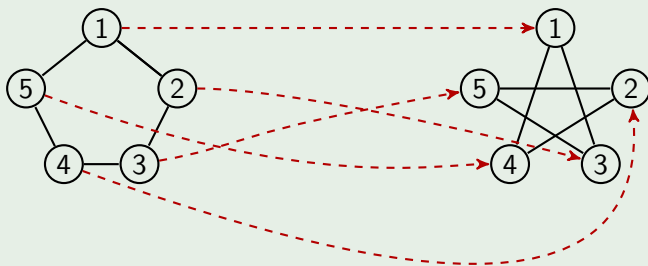
$$h : k \mapsto P_k$$

ein Ordnungshomomorphismus von  $(T_n, |)$  auf  $(\mathcal{P}(P_n), \subseteq)$ .

$h$  ist sogar ein Isomorphismus, falls  $n$  **quadratifrei** ist (d.h. es gibt kein  $k \geq 2$ , so dass  $k^2$  die Zahl  $n$  teilt).

# Isomorphismen

## Beispiel



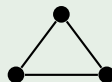
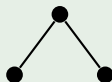
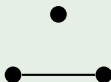
	<hr/>	
	$v$   1 2 3 4 5	
$G = (V, E)$	$h_1(v)$   1 3 5 2 4	$G' = (V, E')$
	$h_2(v)$   1 4 2 5 3	
	<hr/>	

- Die beiden Graphen  $G$  und  $G'$  sind isomorph.
- Zwei Isomorphismen sind beispielsweise  $h_1$  und  $h_2$ .

# Isomorphismen

## Beispiel

- Während auf der Knotenmenge  $V = \{1, 2, 3\}$  insgesamt  $2^{\binom{3}{2}} = 2^3 = 8$  verschiedene Graphen existieren, gibt es auf dieser Menge nur **4** verschiedene nichtisomorphe Graphen:



# Isomorphismen

## Beispiel

- Auf der Knotenmenge  $V = [n]$  existieren genau  $2^{\binom{n}{2}}$  verschiedene Graphen.
- Sei  $a(n)$  die Anzahl aller nichtisomorphen Graphen auf  $V$ .
- Da jede Isomorphieklasse mindestens einen und höchstens  $n!$  Graphen enthält, folgt

$$2^{\binom{n}{2}}/n! \leq a(n) \leq 2^{\binom{n}{2}}.$$

- Tatsächlich ist  $a(n)$  **asymptotisch gleich** der unteren Schranke  $u(n) = 2^{\binom{n}{2}}/n!$  (in Zeichen:  $a(n) \sim u(n)$ ), d.h. es gilt

$$\lim_{n \rightarrow \infty} a(n)/u(n) = 1.$$

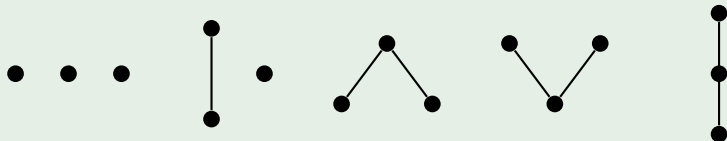
- Also gibt es auf  $V = \{1, \dots, n\}$  nicht wesentlich mehr als  $u(n)$  nichtisomorphe Graphen.



# Isomorphismen

## Beispiel

- Es existieren genau 5 nichtisomorphe Ordnungen mit 3 Elementen:



- Anders ausgedrückt: Die Klasse aller dreielementigen Ordnungen zerfällt unter der Isomorphierelation  $\cong$  in fünf Äquivalenzklassen, die durch obige fünf Hasse-Diagramme repräsentiert werden.

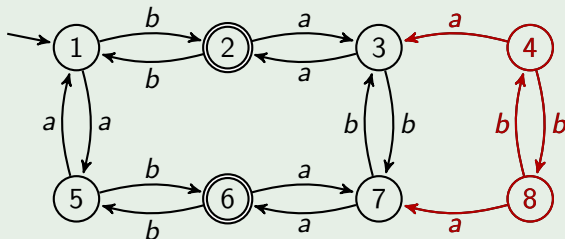
# Minimierung von DFAs

## Frage

Wie können wir feststellen, ob ein DFA  $M = (Z, \Sigma, \delta, q_0, E)$  eine minimale Anzahl von Zuständen besitzt (und  $Z$  evtl. verkleinern)?

## Beispiel

- Betrachte den DFA  $M$



- Zunächst können alle vom Startzustand aus **unerreichbaren Zustände** entfernt werden.

# Minimierung von DFAs

## Frage

Wie können wir feststellen, ob ein DFA  $M = (Z, \Sigma, \delta, q_0, E)$  eine minimale Anzahl von Zuständen besitzt (und  $Z$  evtl. verkleinern)?

## Antwort

- Zunächst können alle vom Startzustand aus unerreichbaren Zustände entfernt werden.
- Zudem lassen sich zwei Zustände  $p$  und  $q$  verschmelzen, wenn  $M$  von  $p$  und  $q$  aus jeweils dieselben Wörter akzeptiert.

- Für  $z \in Z$  sei

$$M_z = (Z, \Sigma, \delta, z, E) \text{ und } L_z = L(M_z).$$

- Dann können wir  $p$  und  $q$  verschmelzen (in Zeichen:  $p \sim q$ ), wenn  $L_p = L_q$  ist.
- Offensichtlich ist  $\sim$  eine Äquivalenzrelation auf  $Z$ .

# Minimierung von DFAs

## Idee

Verschmelze jeden Zustand  $z$  mit allen äquivalenten Zuständen  $z' \sim z$  zu einem neuen Zustand.

## Notation

- Für die durch  $z$  repräsentierte Äquivalenzklasse

$$[z]_{\sim} = \{z' \in Z \mid z' \sim z\} = \{z' \in Z \mid L_{z'} = L_z\}$$

schreiben wir auch einfach  $[z]$  oder  $\tilde{z}$ .

- Für eine Teilmenge  $Q \subseteq Z$  bezeichne

$$\tilde{Q} = \{\tilde{q} \mid q \in Q\}$$

die Menge aller Äquivalenzklassen  $\tilde{q}$ , die mindestens ein Element  $q \in Q$  enthalten.

# Minimierung von DFAs

## Satz

Sei  $M = (Z, \Sigma, \delta, q_0, E)$  ein DFA ohne unerreichbare Zustände. Dann ist

$$\tilde{M} = (\tilde{Z}, \Sigma, \tilde{\delta}, \tilde{q}_0, \tilde{E}) \text{ mit } \tilde{\delta}(\tilde{q}, a) = \widetilde{\delta(q, a)}$$

ein DFA für  $L(M)$  mit einer minimalen Anzahl von Zuständen.

## Beweis

- Zuerst müssen wir zeigen, dass  $\tilde{\delta}$  wohldefiniert ist, also  $\tilde{\delta}(\tilde{q}, a)$  nicht von der Wahl des Repräsentanten  $q$  für die Äquivalenzklasse  $\tilde{q}$  abhängt.
- Hierzu zeigen wir die Implikation  $p \sim q \Rightarrow \delta(p, a) \sim \delta(q, a)$ :

$$\begin{aligned} L_q = L_p &\Leftrightarrow \forall x \in \Sigma^* : x \in L_q \leftrightarrow x \in L_p \\ &\Rightarrow \forall x \in \Sigma^* : ax \in L_q \leftrightarrow ax \in L_p \\ &\Leftrightarrow \forall x \in \Sigma^* : x \in L_{\delta(q,a)} \leftrightarrow x \in L_{\delta(p,a)} \\ &\Leftrightarrow L_{\delta(q,a)} = L_{\delta(p,a)}. \end{aligned}$$

## Minimierung von DFAs

### Satz

Sei  $M = (Z, \Sigma, \delta, q_0, E)$  ein DFA ohne unerreichbare Zustände. Dann ist

$$\tilde{M} = (\tilde{Z}, \Sigma, \tilde{\delta}, \tilde{q}_0, \tilde{E}) \text{ mit } \tilde{\delta}(\tilde{q}, a) = \widetilde{\delta(q, a)}$$

ein DFA für  $L(M)$  mit einer minimalen Anzahl von Zuständen.

### Beweis (Fortsetzung)

- Als nächstes zeigen wir, dass  $L(\tilde{M}) = L(M)$  ist.
- Sei  $x = x_1 \cdots x_n \in \Sigma^*$  und seien  $q_0, q_1, \dots, q_n$  die von  $M$  bei Eingabe  $x$  durchlaufenen Zustände, d.h. es gilt  $\delta(q_{i-1}, x_i) = q_i$  für  $i = 1, \dots, n$ .
- Nach Definition von  $\tilde{\delta}$  folgt daher  $\tilde{\delta}(\tilde{q}_{i-1}, x_i) = \tilde{q}_i$  für  $i = 1, \dots, n$ , d.h.  $\tilde{M}$  durchläuft bei Eingabe  $x$  die Zustände  $\tilde{q}_0, \tilde{q}_1, \dots, \tilde{q}_n$ .
- Da aber  $\tilde{q}_n$  entweder nur End- oder nur Nicht-Endzustände enthält, gehört  $q_n$  genau dann zu  $E$ , wenn  $\tilde{q}_n \in \tilde{E}$ , d.h. es gilt

$$x \in L(M) \Leftrightarrow x \in L(\tilde{M}).$$

# Minimierung von DFAs

## Beweis (Schluss)

- Noch z.z.:  $\tilde{M}$  hat eine minimale Anzahl von Zuständen.
- Da  $\tilde{M}$  nicht mehr Zustände hat als  $M$ , ist  $\tilde{M}$  sicher dann minimal, wenn  $M$  bereits minimal ist.
- Es reicht also zu zeigen, dass die Anzahl  $k = \|\tilde{Z}\| = \|\{L_q \mid q \in Z\}\|$  der Zustände von  $\tilde{M}$  nicht von  $M$ , sondern nur von  $L = L(M)$  abhängt.
- Für  $x \in \Sigma^*$  sei

$$L_x = \{y \in \Sigma^* \mid xy \in L\}.$$

- Dann gilt  $\{L_x \mid x \in \Sigma^*\} = \{L_q \mid q \in Z\}$ :
  - ⊆: Klar, da  $L_x = L_q$  für  $q = \hat{\delta}(q_0, x)$  ist.
  - ⊇: Auch klar, da jedes  $q \in Z$  über ein  $x \in \Sigma^*$  erreichbar ist.
- Also hängt  $k = \|\{L_q \mid q \in Z\}\| = \|\{L_x \mid x \in \Sigma^*\}\|$  nur von  $L$  ab.



## Wie können wir $\tilde{M}$ aus $M$ konstruieren?

Hierzu genügt es, herauszufinden, ob zwei Zustände  $p$  und  $q$  von  $M$  äquivalent sind oder nicht?

- Sei  $A\Delta B = (A \setminus B) \cup (B \setminus A)$  die **symmetrische Differenz** von  $A$  und  $B$ .
- Die Inäquivalenz  $p \not\sim q$  ist also gleichbedeutend mit  $L_p\Delta L_q \neq \emptyset$ .
- Wir nennen ein Wort  $x \in L_p\Delta L_q$  **Unterscheider** zwischen  $p$  und  $q$ .
- Offenbar unterscheidet  $\varepsilon$  Zustände  $p \in E$  von Zuständen  $q \in Z \setminus E$ .
- Falls  $x$  die Zustände  $\delta(p, a)$  und  $\delta(q, a)$  unterscheidet, so unterscheidet  $ax$  die Zustände  $p$  und  $q$ , d.h.  $x \in L_{\delta(p,a)}\Delta L_{\delta(q,a)} \Rightarrow ax \in L_p\Delta L_q$ .
- Wenn also  $D$  nur inäquivalente Zustandspaare enthält, so trifft dies auch auf die Menge

$$D' = \{ \{p, q\} \mid \exists a \in \Sigma : \{\delta(p, a), \delta(q, a)\} \in D \}$$

zu.



# Algorithmische Konstruktion von $\tilde{M}$

## Idee

- Berechne ausgehend von  $D_0 = \{\{p, q\} \mid p \in E, q \notin E\}$  mittels
$$D_{i+1} = D_i \cup \{\{p, q\} \mid \exists a \in \Sigma : \{\delta(p, a), \delta(q, a)\} \in D_i\}$$
eine Folge  $D_0 \subseteq D_1 \subseteq D_2 \subseteq \dots$  von Mengen mit inäquivalenten Zustandspaaren.
- Da es nur endlich viele Zustandspaare gibt, muss es ein  $j$  geben mit  $D_{j+1} = D_j$ .
- Für dieses  $j$  gilt dann

$$p \not\sim q \Leftrightarrow \{p, q\} \in D_j \quad (\text{siehe Übungen}).$$

- Folglich ist

$$\tilde{z} = \{z\} \cup \{z' \in Z \mid \{z, z'\} \notin D_j\}.$$

# Algorithmus zur Berechnung eines minimalen DFA

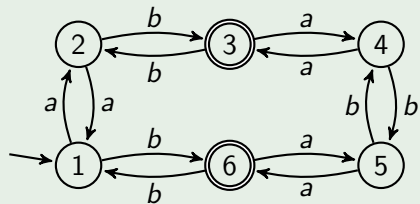
## Algorithmus min-DFA( $M$ )

- 1 **Input:** DFA  $M = (Z, \Sigma, \delta, q_0, E)$
- 2 entferne alle nicht erreichbaren Zustände
- 3  $D' := \{\{z, z'\} \mid z \in E, z' \notin E\}$
- 4 **repeat**
- 5      $D := D'$
- 6      $D' := D \cup \{\{p, q\} \mid \exists a \in \Sigma : \{\delta(p, a), \delta(q, a)\} \in D\}$
- 7 **until**  $D' = D$
- 8 **Output:**  $\tilde{M} = (\tilde{Z}, \Sigma, \tilde{\delta}, \tilde{q}_0, \tilde{E})$ , wobei für jeden Zustand  $z \in Z$  gilt:  $\tilde{z} = \{z\} \cup \{z' \in Z \mid \{z, z'\} \notin D\}$

# Algorithmus für die Konstruktion von $\tilde{M}$

## Beispiel

Betrachte den DFA  $M$



2					
3	$\epsilon$	$\epsilon$			
4			$\epsilon$		
5			$\epsilon$		
6	$\epsilon$	$\epsilon$		$\epsilon$	$\epsilon$
	1	2	3	4	5

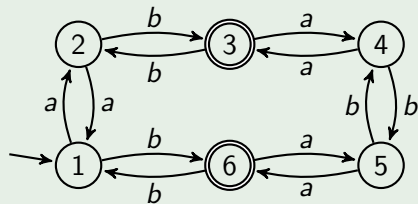
Dann enthält  $D_0$  die Paare

$\{1, 3\}, \{1, 6\}, \{2, 3\}, \{2, 6\}, \{3, 4\}, \{3, 5\}, \{4, 6\}, \{5, 6\}$ .

# Algorithmus für die Konstruktion von $\tilde{M}$

## Beispiel

Betrachte den DFA  $M$



2					
3	$\epsilon$	$\epsilon$			
4	$a$	$a$	$\epsilon$		
5	$a$	$a$	$\epsilon$		
6	$\epsilon$	$\epsilon$		$\epsilon$	$\epsilon$
	1	2	3	4	5

Wegen

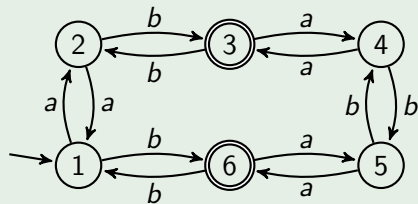
$\{p, q\}$	$\{1, 4\}$	$\{1, 5\}$	$\{2, 4\}$	$\{2, 5\}$
$\{\delta(q, a), \delta(p, a)\}$	$\{2, 3\}$	$\{2, 6\}$	$\{1, 3\}$	$\{1, 6\}$

enthält  $D_1$  zusätzlich die Paare  $\{1, 4\}$ ,  $\{1, 5\}$ ,  $\{2, 4\}$ ,  $\{2, 5\}$ .

# Algorithmus für die Konstruktion von $\tilde{M}$

## Beispiel

Betrachte den DFA  $M$



2					
3	$\epsilon$	$\epsilon$			
4	$a$	$a$	$\epsilon$		
5	$a$	$a$	$\epsilon$		
6	$\epsilon$	$\epsilon$		$\epsilon$	$\epsilon$
	1	2	3	4	5

Da nun jedoch die verbliebenen Paare  $\{1, 2\}$ ,  $\{3, 6\}$ ,  $\{4, 5\}$  wegen

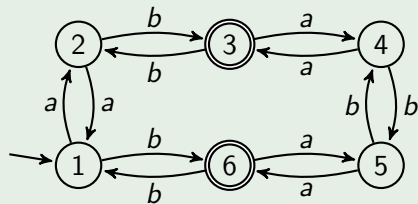
$\{p, q\}$	$\{1, 2\}$	$\{3, 6\}$	$\{4, 5\}$
$\{\delta(p, a), \delta(q, a)\}$	$\{1, 2\}$	$\{4, 5\}$	$\{3, 6\}$
$\{\delta(p, b), \delta(q, b)\}$	$\{3, 6\}$	$\{1, 2\}$	$\{4, 5\}$

nicht zu  $D_1$  hinzugefügt werden können, ist  $D_2 = D_1$ .

# Algorithmus für die Konstruktion von $\tilde{M}$

## Beispiel

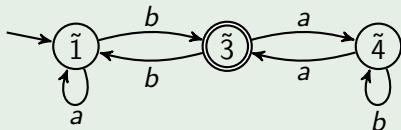
Betrachte den DFA  $M$



2					
3	$\epsilon$	$\epsilon$			
4	$a$	$a$	$\epsilon$		
5	$a$	$a$	$\epsilon$		
6	$\epsilon$	$\epsilon$		$\epsilon$	$\epsilon$
	1	2	3	4	5

Da die Paare  $\{1, 2\}$ ,  $\{3, 6\}$  und  $\{4, 5\}$  nicht in  $D_1$  enthalten sind, können die Zustände 1 und 2, 3 und 6, sowie 4 und 5 verschmolzen werden.

Demnach hat  $\tilde{M}$  die Zustände  $\tilde{1} = \{1, 2\}$ ,  $\tilde{3} = \{3, 6\}$  und  $\tilde{4} = \{4, 5\}$ :



# Direkte Konstruktion eines Minimal-DFA aus $L$

## Bemerkung

- $\tilde{M}$  erreicht nach Lesen von  $x$  den Zustand  $\widehat{\delta}(q_0, x)$ . Wegen

$$\begin{aligned}\widehat{\delta}(q_0, x) = \widehat{\delta}(q_0, y) &\Leftrightarrow \widehat{\delta}(q_0, x) \sim \widehat{\delta}(q_0, y) \\ &\Leftrightarrow L_{\widehat{\delta}(q_0, x)} = L_{\widehat{\delta}(q_0, y)} \Leftrightarrow L_x = L_y\end{aligned}$$

können die Zustände  $\widehat{\delta}(q_0, x)$  von  $\tilde{M}$  auch mit  $L_x$  bezeichnet werden.

- Dies führt auf den zu  $\tilde{M}$  isomorphen DFA  $M_L = (Z_L, \Sigma, \delta_L, L_\epsilon, E_L)$  mit

$$Z_L = \{L_x \mid x \in \Sigma^*\}, \quad E_L = \{L_x \mid x \in L\} \text{ und } \delta_L(L_x, a) = L_{xa},$$

der sich auch direkt aus der Sprache  $L$  gewinnen lässt.

- Notwendig und hinreichend für die Existenz von  $M_L$  ist, dass die Menge  $Z_L = \{L_x \mid x \in \Sigma^*\}$  endlich ist.
- $L$  ist also genau dann regulär, wenn der Index der durch

$$x R_L y \Leftrightarrow L_x = L_y$$

auf  $\Sigma^*$  definierten Äquivalenzrelation  $R_L$  endlich ist.

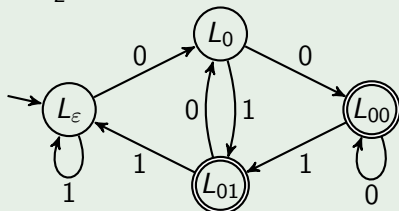
# Direkte Konstruktion eines Minimal-DFA aus $L$

## Beispiel

- Betrachte die Sprache  $L = \{x_1 \cdots x_n \in \{0, 1\}^* \mid x_{n-1} = 0\}$ .
- Dann hat  $M_L$  die folgenden Sprachen als Zustände:

$$L_x = \begin{cases} L, & x \in \{\varepsilon, 1\} \text{ oder } x \text{ endet mit } 11, \\ L \cup \{0, 1\}, & x = 0 \text{ oder } x \text{ endet mit } 10, \\ L \cup \{\varepsilon, 0, 1\}, & x \text{ endet mit } 00, \\ L \cup \{\varepsilon\}, & x \text{ endet mit } 01. \end{cases}$$

- Graphische Darstellung von  $M_L$ :





## Der Satz von Myhill und Nerode

- Ist  $M$  ein DFA mit einer minimalen Anzahl von Zuständen, so haben die Zustände von  $\tilde{M}$  die Form  $\tilde{q} = \{q\}$ , d.h.  $M$  ist isomorph zu  $\tilde{M}$ .
- Da  $\tilde{M}$  wiederum isomorph zu  $M_L$  ist, ist jeder minimale DFA  $M$  mit  $L(M) = L$  isomorph zu  $M_L$ , d.h. für jede reguläre Sprache  $L$  gibt es bis auf Isomorphie nur einen Minimal-DFA.

### Satz (Myhill und Nerode)

Für eine Sprache  $L \subseteq \Sigma^*$  sei

$$\begin{aligned} R_L &= \{(x, y) \in \Sigma^* \times \Sigma^* \mid L_x = L_y\} \\ &= \{(x, y) \in \Sigma^* \times \Sigma^* \mid \forall z \in \Sigma^* : xz \in L \Leftrightarrow yz \in L\} \end{aligned}$$

und sei  $\text{index}(R_L)$  der Index von  $R_L$ . Dann gilt:

- 1 REG =  $\{L \mid \text{index}(R_L) < \infty\}$ .
- 2 Für jede reguläre Sprache  $L$  gibt es bis auf Isomorphie genau einen Minimal-DFA. Dieser hat  $\text{index}(R_L)$  Zustände.

## Der Äquivalenzklassen-DFA $M_{R_L}$ für $L$

- Zwei Eingaben  $x$  und  $y$  überführen den DFA  $M_L$  genau dann in denselben Zustand, wenn  $L_x = L_y$  ist (also  $xR_L y$  gilt).
- Die Zustände von  $M_L$  können daher anstelle von  $L_x$  auch mit den Äquivalenzklassen  $[x]$  von  $R_L$  (bzw. mit geeigneten Repräsentanten) benannt werden.
- Der resultierende Minimal-DFA  $M_{R_L}$  wird auch als **Äquivalenzklassen-automat** bezeichnet:

$$M_{R_L} = (Z, \Sigma, \delta, [\varepsilon], E) \text{ mit } Z = \{[x] \mid x \in \Sigma^*\} \text{ und } E = \{[x] \mid x \in L\}.$$

- Für die Konstruktion von  $\delta$  genügt es, ausgehend von  $r_1 = \varepsilon$  eine Folge von Wörtern  $r_1, \dots, r_k$  mit  $[r_i] \neq [r_j]$  zu bestimmen, so dass zu jedem  $r_i$  und jedem Zeichen  $a \in \Sigma$  ein  $r_j$  existiert mit  $r_i a \in [r_j]$ .
- In diesem Fall ist dann  $\delta([r_i], a) = [r_i a] = [r_j]$ .
- Die Konstruktion von  $M_{R_L}$  erfordert meist weniger Aufwand als die von  $M_L$ , da die Bestimmung der Sprachen  $L_x$  entfällt.

# Direkte Konstruktion des Äquivalenzklassen-DFA $M_{R_L}$ aus $L$

## Beispiel

Für die Sprache  $L = \{x_1 \cdots x_n \in \{0, 1\}^* \mid x_{n-1} = 0\}$

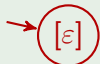
$r$	
$[r0]$	
$[r1]$	

# Direkte Konstruktion des Äquivalenzklassen-DFA $M_{R_L}$ aus $L$

## Beispiel

Für die Sprache  $L = \{x_1 \cdots x_n \in \{0, 1\}^* \mid x_{n-1} = 0\}$  lässt sich  $M_{R_L}$  ausgehend von  $r_1 = \varepsilon$  wie folgt konstruieren:

$r$	$\varepsilon$
$[r0]$	
$[r1]$	



# Direkte Konstruktion des Äquivalenzklassen-DFA $M_{R_L}$ aus $L$

## Beispiel

Für die Sprache  $L = \{x_1 \cdots x_n \in \{0, 1\}^* \mid x_{n-1} = 0\}$  lässt sich  $M_{R_L}$  ausgehend von  $r_1 = \varepsilon$  wie folgt konstruieren:

- 1 Wegen  $r_1 0 = 0$

$r$	$\varepsilon$
$[r0]$	
$[r1]$	



# Direkte Konstruktion des Äquivalenzklassen-DFA $M_{R_L}$ aus $L$

## Beispiel

Für die Sprache  $L = \{x_1 \cdots x_n \in \{0, 1\}^* \mid x_{n-1} = 0\}$  lässt sich  $M_{R_L}$  ausgehend von  $r_1 = \varepsilon$  wie folgt konstruieren:

- 1 Wegen  $r_1 0 = 0 \notin [\varepsilon]$

$r$	$\varepsilon$
$[r0]$	
$[r1]$	



# Direkte Konstruktion des Äquivalenzklassen-DFA $M_{R_L}$ aus $L$

## Beispiel

Für die Sprache  $L = \{x_1 \cdots x_n \in \{0, 1\}^* \mid x_{n-1} = 0\}$  lässt sich  $M_{R_L}$  ausgehend von  $r_1 = \varepsilon$  wie folgt konstruieren:

- 1 Wegen  $r_1 0 = 0 \notin [\varepsilon]$  ist  $r_2 = 0$  und

$r$	$\varepsilon$	$0$
$[r0]$		
$[r1]$		



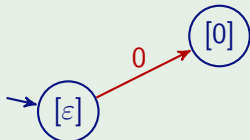
# Direkte Konstruktion des Äquivalenzklassen-DFA $M_{R_L}$ aus $L$

## Beispiel

Für die Sprache  $L = \{x_1 \cdots x_n \in \{0, 1\}^* \mid x_{n-1} = 0\}$  lässt sich  $M_{R_L}$  ausgehend von  $r_1 = \varepsilon$  wie folgt konstruieren:

- 1 Wegen  $r_1 0 = 0 \notin [\varepsilon]$  ist  $r_2 = 0$  und  $\delta([\varepsilon], 0) = [0]$ .

$r$	$\varepsilon$	$0$
$[r0]$	$[0]$	
$[r1]$		





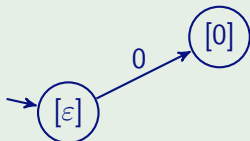
# Direkte Konstruktion des Äquivalenzklassen-DFA $M_{R_L}$ aus $L$

## Beispiel

Für die Sprache  $L = \{x_1 \cdots x_n \in \{0, 1\}^* \mid x_{n-1} = 0\}$  lässt sich  $M_{R_L}$  ausgehend von  $r_1 = \varepsilon$  wie folgt konstruieren:

- 1 Wegen  $r_1 0 = 0 \notin [\varepsilon]$  ist  $r_2 = 0$  und  $\delta([\varepsilon], 0) = [0]$ .
- 2 Wegen  $r_1 1 = 1$

$r$	$\varepsilon$	$0$
$[r0]$	$[0]$	
$[r1]$		



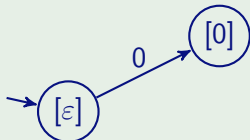
# Direkte Konstruktion des Äquivalenzklassen-DFA $M_{R_L}$ aus $L$

## Beispiel

Für die Sprache  $L = \{x_1 \cdots x_n \in \{0, 1\}^* \mid x_{n-1} = 0\}$  lässt sich  $M_{R_L}$  ausgehend von  $r_1 = \varepsilon$  wie folgt konstruieren:

- 1 Wegen  $r_1 0 = 0 \notin [\varepsilon]$  ist  $r_2 = 0$  und  $\delta([\varepsilon], 0) = [0]$ .
- 2 Wegen  $r_1 1 = 1 \in [\varepsilon]$

$r$	$\varepsilon$	$0$
$[r0]$	$[0]$	
$[r1]$		



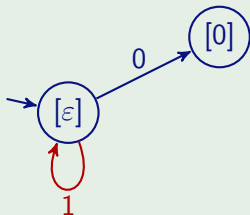
# Direkte Konstruktion des Äquivalenzklassen-DFA $M_{R_L}$ aus $L$

## Beispiel

Für die Sprache  $L = \{x_1 \cdots x_n \in \{0, 1\}^* \mid x_{n-1} = 0\}$  lässt sich  $M_{R_L}$  ausgehend von  $r_1 = \varepsilon$  wie folgt konstruieren:

- 1 Wegen  $r_1 0 = 0 \notin [\varepsilon]$  ist  $r_2 = 0$  und  $\delta([\varepsilon], 0) = [0]$ .
- 2 Wegen  $r_1 1 = 1 \in [\varepsilon]$  ist  $\delta([\varepsilon], 1) = [\varepsilon]$ .

$r$	$\varepsilon$	$0$
$[r0]$	$[0]$	
$[r1]$	$[\varepsilon]$	



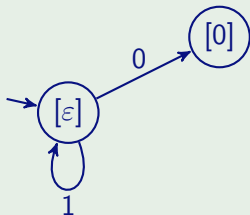
# Direkte Konstruktion des Äquivalenzklassen-DFA $M_{R_L}$ aus $L$

## Beispiel

Für die Sprache  $L = \{x_1 \cdots x_n \in \{0, 1\}^* \mid x_{n-1} = 0\}$  lässt sich  $M_{R_L}$  ausgehend von  $r_1 = \varepsilon$  wie folgt konstruieren:

- 1 Wegen  $r_1 0 = 0 \notin [\varepsilon]$  ist  $r_2 = 0$  und  $\delta([\varepsilon], 0) = [0]$ .
- 2 Wegen  $r_1 1 = 1 \in [\varepsilon]$  ist  $\delta([\varepsilon], 1) = [\varepsilon]$ .
- 3 Wegen  $r_2 0 = 00$

$r$	$\varepsilon$	$0$
$[r0]$	$[0]$	
$[r1]$	$[\varepsilon]$	



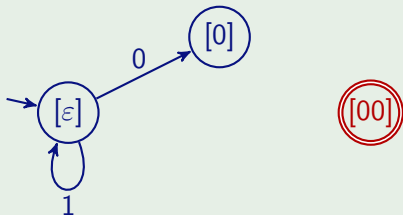
# Direkte Konstruktion des Äquivalenzklassen-DFA $M_{R_L}$ aus $L$

## Beispiel

Für die Sprache  $L = \{x_1 \cdots x_n \in \{0, 1\}^* \mid x_{n-1} = 0\}$  lässt sich  $M_{R_L}$  ausgehend von  $r_1 = \varepsilon$  wie folgt konstruieren:

- 1 Wegen  $r_1 0 = 0 \notin [\varepsilon]$  ist  $r_2 = 0$  und  $\delta([\varepsilon], 0) = [0]$ .
- 2 Wegen  $r_1 1 = 1 \in [\varepsilon]$  ist  $\delta([\varepsilon], 1) = [\varepsilon]$ .
- 3 Wegen  $r_2 0 = 00 \notin [\varepsilon] \cup [0]$  ist  $r_3 = 00$  und

$r$	$\varepsilon$	0	00
$[r0]$	$[0]$		
$[r1]$	$[\varepsilon]$		



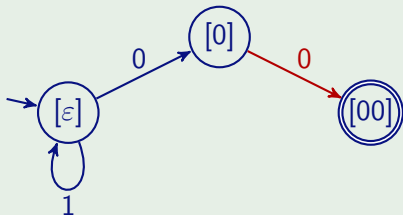
# Direkte Konstruktion des Äquivalenzklassen-DFA $M_{R_L}$ aus $L$

## Beispiel

Für die Sprache  $L = \{x_1 \cdots x_n \in \{0, 1\}^* \mid x_{n-1} = 0\}$  lässt sich  $M_{R_L}$  ausgehend von  $r_1 = \varepsilon$  wie folgt konstruieren:

- 1 Wegen  $r_1 0 = 0 \notin [\varepsilon]$  ist  $r_2 = 0$  und  $\delta([\varepsilon], 0) = [0]$ .
- 2 Wegen  $r_1 1 = 1 \in [\varepsilon]$  ist  $\delta([\varepsilon], 1) = [\varepsilon]$ .
- 3 Wegen  $r_2 0 = 00 \notin [\varepsilon] \cup [0]$  ist  $r_3 = 00$  und  $\delta([0], 0) = [00]$ .

$r$	$\varepsilon$	0	00
$[r0]$	$[0]$	$[00]$	
$[r1]$	$[\varepsilon]$		



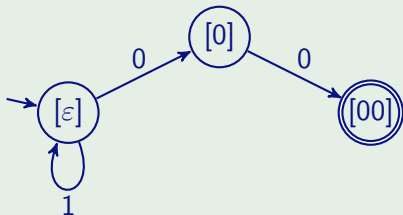
# Direkte Konstruktion des Äquivalenzklassen-DFA $M_{R_L}$ aus $L$

## Beispiel

Für die Sprache  $L = \{x_1 \cdots x_n \in \{0, 1\}^* \mid x_{n-1} = 0\}$  lässt sich  $M_{R_L}$  ausgehend von  $r_1 = \varepsilon$  wie folgt konstruieren:

- 1 Wegen  $r_1 0 = 0 \notin [\varepsilon]$  ist  $r_2 = 0$  und  $\delta([\varepsilon], 0) = [0]$ .
- 2 Wegen  $r_1 1 = 1 \in [\varepsilon]$  ist  $\delta([\varepsilon], 1) = [\varepsilon]$ .
- 3 Wegen  $r_2 0 = 00 \notin [\varepsilon] \cup [0]$  ist  $r_3 = 00$  und  $\delta([0], 0) = [00]$ .
- 4 Wegen  $r_2 1 = 01$

$r$	$\varepsilon$	0	00
$[r0]$	$[0]$	$[00]$	
$[r1]$	$[\varepsilon]$		



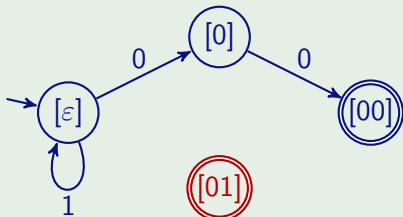
# Direkte Konstruktion des Äquivalenzklassen-DFA $M_{R_L}$ aus $L$

## Beispiel

Für die Sprache  $L = \{x_1 \cdots x_n \in \{0, 1\}^* \mid x_{n-1} = 0\}$  lässt sich  $M_{R_L}$  ausgehend von  $r_1 = \varepsilon$  wie folgt konstruieren:

- 1 Wegen  $r_1 0 = 0 \notin [\varepsilon]$  ist  $r_2 = 0$  und  $\delta([\varepsilon], 0) = [0]$ .
- 2 Wegen  $r_1 1 = 1 \in [\varepsilon]$  ist  $\delta([\varepsilon], 1) = [\varepsilon]$ .
- 3 Wegen  $r_2 0 = 00 \notin [\varepsilon] \cup [0]$  ist  $r_3 = 00$  und  $\delta([0], 0) = [00]$ .
- 4 Wegen  $r_2 1 = 01 \notin [\varepsilon] \cup [0] \cup [00]$  ist  $r_4 = 01$  und

$r$	$\varepsilon$	0	00	01
$[r0]$	$[0]$	$[00]$		
$[r1]$	$[\varepsilon]$			





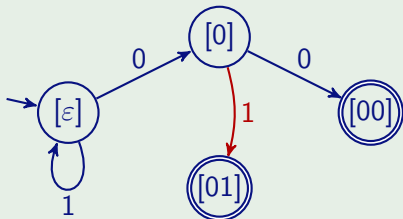
# Direkte Konstruktion des Äquivalenzklassen-DFA $M_{R_L}$ aus $L$

## Beispiel

Für die Sprache  $L = \{x_1 \cdots x_n \in \{0, 1\}^* \mid x_{n-1} = 0\}$  lässt sich  $M_{R_L}$  ausgehend von  $r_1 = \varepsilon$  wie folgt konstruieren:

- 1 Wegen  $r_1 0 = 0 \notin [\varepsilon]$  ist  $r_2 = 0$  und  $\delta([\varepsilon], 0) = [0]$ .
- 2 Wegen  $r_1 1 = 1 \in [\varepsilon]$  ist  $\delta([\varepsilon], 1) = [\varepsilon]$ .
- 3 Wegen  $r_2 0 = 00 \notin [\varepsilon] \cup [0]$  ist  $r_3 = 00$  und  $\delta([0], 0) = [00]$ .
- 4 Wegen  $r_2 1 = 01 \notin [\varepsilon] \cup [0] \cup [00]$  ist  $r_4 = 01$  und  $\delta([0], 1) = [01]$ .

$r$	$\varepsilon$	0	00	01
$[r0]$	$[0]$	$[00]$		
$[r1]$	$[\varepsilon]$	$[01]$		



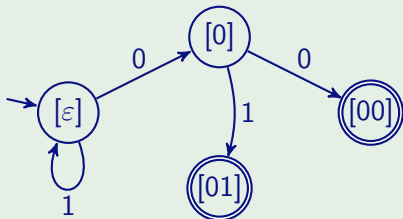
# Direkte Konstruktion des Äquivalenzklassen-DFA $M_{R_L}$ aus $L$

## Beispiel

Für die Sprache  $L = \{x_1 \cdots x_n \in \{0, 1\}^* \mid x_{n-1} = 0\}$  lässt sich  $M_{R_L}$  ausgehend von  $r_1 = \varepsilon$  wie folgt konstruieren:

- 1 Wegen  $r_1 0 = 0 \notin [\varepsilon]$  ist  $r_2 = 0$  und  $\delta([\varepsilon], 0) = [0]$ .
- 2 Wegen  $r_1 1 = 1 \in [\varepsilon]$  ist  $\delta([\varepsilon], 1) = [\varepsilon]$ .
- 3 Wegen  $r_2 0 = 00 \notin [\varepsilon] \cup [0]$  ist  $r_3 = 00$  und  $\delta([0], 0) = [00]$ .
- 4 Wegen  $r_2 1 = 01 \notin [\varepsilon] \cup [0] \cup [00]$  ist  $r_4 = 01$  und  $\delta([0], 1) = [01]$ .
- 5 Wegen  $r_3 0 = 000$

$r$	$\varepsilon$	0	00	01
$[r0]$	$[0]$	$[00]$		
$[r1]$	$[\varepsilon]$	$[01]$		



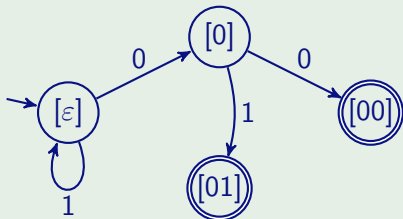
# Direkte Konstruktion des Äquivalenzklassen-DFA $M_{R_L}$ aus $L$

## Beispiel

Für die Sprache  $L = \{x_1 \cdots x_n \in \{0, 1\}^* \mid x_{n-1} = 0\}$  lässt sich  $M_{R_L}$  ausgehend von  $r_1 = \varepsilon$  wie folgt konstruieren:

- 1 Wegen  $r_1 0 = 0 \notin [\varepsilon]$  ist  $r_2 = 0$  und  $\delta([\varepsilon], 0) = [0]$ .
- 2 Wegen  $r_1 1 = 1 \in [\varepsilon]$  ist  $\delta([\varepsilon], 1) = [\varepsilon]$ .
- 3 Wegen  $r_2 0 = 00 \notin [\varepsilon] \cup [0]$  ist  $r_3 = 00$  und  $\delta([0], 0) = [00]$ .
- 4 Wegen  $r_2 1 = 01 \notin [\varepsilon] \cup [0] \cup [00]$  ist  $r_4 = 01$  und  $\delta([0], 1) = [01]$ .
- 5 Wegen  $r_3 0 = 000 \in [00]$

$r$	$\varepsilon$	0	00	01
$[r0]$	$[0]$	$[00]$		
$[r1]$	$[\varepsilon]$	$[01]$		



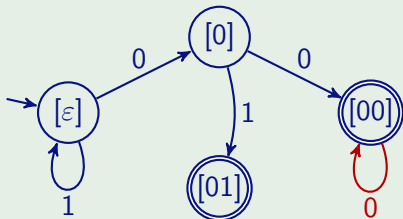
# Direkte Konstruktion des Äquivalenzklassen-DFA $M_{R_L}$ aus $L$

## Beispiel

Für die Sprache  $L = \{x_1 \cdots x_n \in \{0, 1\}^* \mid x_{n-1} = 0\}$  lässt sich  $M_{R_L}$  ausgehend von  $r_1 = \varepsilon$  wie folgt konstruieren:

- 1 Wegen  $r_1 0 = 0 \notin [\varepsilon]$  ist  $r_2 = 0$  und  $\delta([\varepsilon], 0) = [0]$ .
- 2 Wegen  $r_1 1 = 1 \in [\varepsilon]$  ist  $\delta([\varepsilon], 1) = [\varepsilon]$ .
- 3 Wegen  $r_2 0 = 00 \notin [\varepsilon] \cup [0]$  ist  $r_3 = 00$  und  $\delta([0], 0) = [00]$ .
- 4 Wegen  $r_2 1 = 01 \notin [\varepsilon] \cup [0] \cup [00]$  ist  $r_4 = 01$  und  $\delta([0], 1) = [01]$ .
- 5 Wegen  $r_3 0 = 000 \in [00]$  ist  $\delta([00], 0) = [00]$ .

$r$	$\varepsilon$	0	00	01
$[r0]$	$[0]$	$[00]$	$[00]$	
$[r1]$	$[\varepsilon]$	$[01]$		



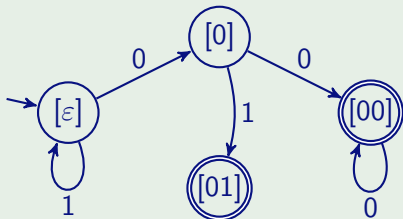
# Direkte Konstruktion des Äquivalenzklassen-DFA $M_{R_L}$ aus $L$

## Beispiel

Für die Sprache  $L = \{x_1 \cdots x_n \in \{0, 1\}^* \mid x_{n-1} = 0\}$  lässt sich  $M_{R_L}$  ausgehend von  $r_1 = \varepsilon$  wie folgt konstruieren:

- 1 Wegen  $r_1 0 = 0 \notin [\varepsilon]$  ist  $r_2 = 0$  und  $\delta([\varepsilon], 0) = [0]$ .
- 2 Wegen  $r_1 1 = 1 \in [\varepsilon]$  ist  $\delta([\varepsilon], 1) = [\varepsilon]$ .
- 3 Wegen  $r_2 0 = 00 \notin [\varepsilon] \cup [0]$  ist  $r_3 = 00$  und  $\delta([0], 0) = [00]$ .
- 4 Wegen  $r_2 1 = 01 \notin [\varepsilon] \cup [0] \cup [00]$  ist  $r_4 = 01$  und  $\delta([0], 1) = [01]$ .
- 5 Wegen  $r_3 0 = 000 \in [00]$  ist  $\delta([00], 0) = [00]$ .
- 6 Wegen  $r_3 1 = 001$

$r$	$\varepsilon$	0	00	01
$[r0]$	$[0]$	$[00]$	$[00]$	
$[r1]$	$[\varepsilon]$	$[01]$		



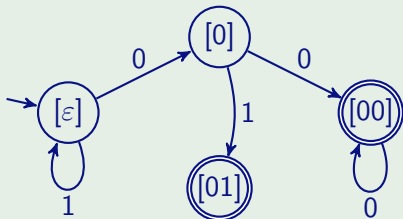
# Direkte Konstruktion des Äquivalenzklassen-DFA $M_{R_L}$ aus $L$

## Beispiel

Für die Sprache  $L = \{x_1 \cdots x_n \in \{0, 1\}^* \mid x_{n-1} = 0\}$  lässt sich  $M_{R_L}$  ausgehend von  $r_1 = \varepsilon$  wie folgt konstruieren:

- 1 Wegen  $r_1 0 = 0 \notin [\varepsilon]$  ist  $r_2 = 0$  und  $\delta([\varepsilon], 0) = [0]$ .
- 2 Wegen  $r_1 1 = 1 \in [\varepsilon]$  ist  $\delta([\varepsilon], 1) = [\varepsilon]$ .
- 3 Wegen  $r_2 0 = 00 \notin [\varepsilon] \cup [0]$  ist  $r_3 = 00$  und  $\delta([0], 0) = [00]$ .
- 4 Wegen  $r_2 1 = 01 \notin [\varepsilon] \cup [0] \cup [00]$  ist  $r_4 = 01$  und  $\delta([0], 1) = [01]$ .
- 5 Wegen  $r_3 0 = 000 \in [00]$  ist  $\delta([00], 0) = [00]$ .
- 6 Wegen  $r_3 1 = 001 \in [01]$

$r$	$\varepsilon$	0	00	01
$[r0]$	$[0]$	$[00]$	$[00]$	
$[r1]$	$[\varepsilon]$	$[01]$		



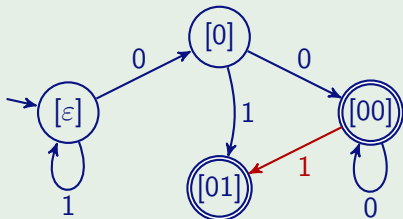
# Direkte Konstruktion des Äquivalenzklassen-DFA $M_{R_L}$ aus $L$

## Beispiel

Für die Sprache  $L = \{x_1 \cdots x_n \in \{0, 1\}^* \mid x_{n-1} = 0\}$  lässt sich  $M_{R_L}$  ausgehend von  $r_1 = \varepsilon$  wie folgt konstruieren:

- 1 Wegen  $r_1 0 = 0 \notin [\varepsilon]$  ist  $r_2 = 0$  und  $\delta([\varepsilon], 0) = [0]$ .
- 2 Wegen  $r_1 1 = 1 \in [\varepsilon]$  ist  $\delta([\varepsilon], 1) = [\varepsilon]$ .
- 3 Wegen  $r_2 0 = 00 \notin [\varepsilon] \cup [0]$  ist  $r_3 = 00$  und  $\delta([0], 0) = [00]$ .
- 4 Wegen  $r_2 1 = 01 \notin [\varepsilon] \cup [0] \cup [00]$  ist  $r_4 = 01$  und  $\delta([0], 1) = [01]$ .
- 5 Wegen  $r_3 0 = 000 \in [00]$  ist  $\delta([00], 0) = [00]$ .
- 6 Wegen  $r_3 1 = 001 \in [01]$  ist  $\delta([00], 1) = [01]$ .

$r$	$\varepsilon$	0	00	01
$[r0]$	$[0]$	$[00]$	$[00]$	
$[r1]$	$[\varepsilon]$	$[01]$	$[01]$	



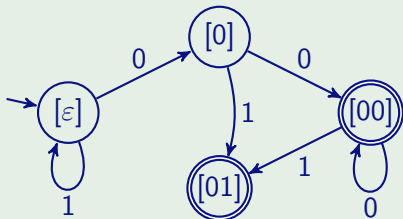
# Direkte Konstruktion des Äquivalenzklassen-DFA $M_{R_L}$ aus $L$

## Beispiel

Für die Sprache  $L = \{x_1 \cdots x_n \in \{0, 1\}^* \mid x_{n-1} = 0\}$  lässt sich  $M_{R_L}$  ausgehend von  $r_1 = \varepsilon$  wie folgt konstruieren:

- 1 Wegen  $r_1 0 = 0 \notin [\varepsilon]$  ist  $r_2 = 0$  und  $\delta([\varepsilon], 0) = [0]$ .
- 2 Wegen  $r_1 1 = 1 \in [\varepsilon]$  ist  $\delta([\varepsilon], 1) = [\varepsilon]$ .
- 3 Wegen  $r_2 0 = 00 \notin [\varepsilon] \cup [0]$  ist  $r_3 = 00$  und  $\delta([0], 0) = [00]$ .
- 4 Wegen  $r_2 1 = 01 \notin [\varepsilon] \cup [0] \cup [00]$  ist  $r_4 = 01$  und  $\delta([0], 1) = [01]$ .
- 5 Wegen  $r_3 0 = 000 \in [00]$  ist  $\delta([00], 0) = [00]$ .
- 6 Wegen  $r_3 1 = 001 \in [01]$  ist  $\delta([00], 1) = [01]$ .
- 7 Wegen  $r_4 0 = 010$

$r$	$\varepsilon$	0	00	01
$[r0]$	$[0]$	$[00]$	$[00]$	
$[r1]$	$[\varepsilon]$	$[01]$	$[01]$	





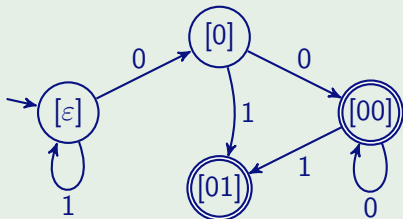
# Direkte Konstruktion des Äquivalenzklassen-DFA $M_{R_L}$ aus $L$

## Beispiel

Für die Sprache  $L = \{x_1 \cdots x_n \in \{0, 1\}^* \mid x_{n-1} = 0\}$  lässt sich  $M_{R_L}$  ausgehend von  $r_1 = \varepsilon$  wie folgt konstruieren:

- 1 Wegen  $r_1 0 = 0 \notin [\varepsilon]$  ist  $r_2 = 0$  und  $\delta([\varepsilon], 0) = [0]$ .
- 2 Wegen  $r_1 1 = 1 \in [\varepsilon]$  ist  $\delta([\varepsilon], 1) = [\varepsilon]$ .
- 3 Wegen  $r_2 0 = 00 \notin [\varepsilon] \cup [0]$  ist  $r_3 = 00$  und  $\delta([0], 0) = [00]$ .
- 4 Wegen  $r_2 1 = 01 \notin [\varepsilon] \cup [0] \cup [00]$  ist  $r_4 = 01$  und  $\delta([0], 1) = [01]$ .
- 5 Wegen  $r_3 0 = 000 \in [00]$  ist  $\delta([00], 0) = [00]$ .
- 6 Wegen  $r_3 1 = 001 \in [01]$  ist  $\delta([00], 1) = [01]$ .
- 7 Wegen  $r_4 0 = 010 \in [0]$

$r$	$\varepsilon$	0	00	01
$[r0]$	$[0]$	$[00]$	$[00]$	
$[r1]$	$[\varepsilon]$	$[01]$	$[01]$	



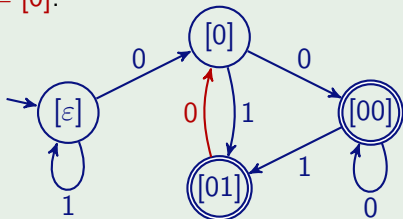
# Direkte Konstruktion des Äquivalenzklassen-DFA $M_{R_L}$ aus $L$

## Beispiel

Für die Sprache  $L = \{x_1 \cdots x_n \in \{0, 1\}^* \mid x_{n-1} = 0\}$  lässt sich  $M_{R_L}$  ausgehend von  $r_1 = \varepsilon$  wie folgt konstruieren:

- 1 Wegen  $r_1 0 = 0 \notin [\varepsilon]$  ist  $r_2 = 0$  und  $\delta([\varepsilon], 0) = [0]$ .
- 2 Wegen  $r_1 1 = 1 \in [\varepsilon]$  ist  $\delta([\varepsilon], 1) = [\varepsilon]$ .
- 3 Wegen  $r_2 0 = 00 \notin [\varepsilon] \cup [0]$  ist  $r_3 = 00$  und  $\delta([0], 0) = [00]$ .
- 4 Wegen  $r_2 1 = 01 \notin [\varepsilon] \cup [0] \cup [00]$  ist  $r_4 = 01$  und  $\delta([0], 1) = [01]$ .
- 5 Wegen  $r_3 0 = 000 \in [00]$  ist  $\delta([00], 0) = [00]$ .
- 6 Wegen  $r_3 1 = 001 \in [01]$  ist  $\delta([00], 1) = [01]$ .
- 7 Wegen  $r_4 0 = 010 \in [0]$  ist  $\delta([01], 0) = [0]$ .

$r$	$\varepsilon$	0	00	01
$[r0]$	$[0]$	$[00]$	$[00]$	$[0]$
$[r1]$	$[\varepsilon]$	$[01]$	$[01]$	



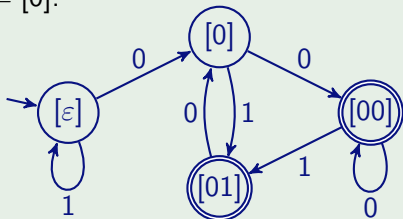
# Direkte Konstruktion des Äquivalenzklassen-DFA $M_{R_L}$ aus $L$

## Beispiel

Für die Sprache  $L = \{x_1 \cdots x_n \in \{0, 1\}^* \mid x_{n-1} = 0\}$  lässt sich  $M_{R_L}$  ausgehend von  $r_1 = \varepsilon$  wie folgt konstruieren:

- 1 Wegen  $r_1 0 = 0 \notin [\varepsilon]$  ist  $r_2 = 0$  und  $\delta([\varepsilon], 0) = [0]$ .
- 2 Wegen  $r_1 1 = 1 \in [\varepsilon]$  ist  $\delta([\varepsilon], 1) = [\varepsilon]$ .
- 3 Wegen  $r_2 0 = 00 \notin [\varepsilon] \cup [0]$  ist  $r_3 = 00$  und  $\delta([0], 0) = [00]$ .
- 4 Wegen  $r_2 1 = 01 \notin [\varepsilon] \cup [0] \cup [00]$  ist  $r_4 = 01$  und  $\delta([0], 1) = [01]$ .
- 5 Wegen  $r_3 0 = 000 \in [00]$  ist  $\delta([00], 0) = [00]$ .
- 6 Wegen  $r_3 1 = 001 \in [01]$  ist  $\delta([00], 1) = [01]$ .
- 7 Wegen  $r_4 0 = 010 \in [0]$  ist  $\delta([01], 0) = [0]$ .
- 8 Wegen  $r_4 1 = 011$

$r$	$\varepsilon$	0	00	01
$[r0]$	$[0]$	$[00]$	$[00]$	$[0]$
$[r1]$	$[\varepsilon]$	$[01]$	$[01]$	



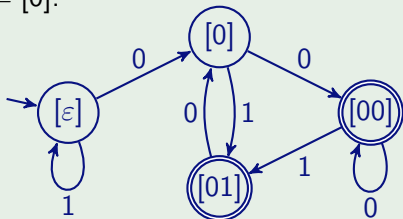
# Direkte Konstruktion des Äquivalenzklassen-DFA $M_{R_L}$ aus $L$

## Beispiel

Für die Sprache  $L = \{x_1 \cdots x_n \in \{0, 1\}^* \mid x_{n-1} = 0\}$  lässt sich  $M_{R_L}$  ausgehend von  $r_1 = \varepsilon$  wie folgt konstruieren:

- 1 Wegen  $r_1 0 = 0 \notin [\varepsilon]$  ist  $r_2 = 0$  und  $\delta([\varepsilon], 0) = [0]$ .
- 2 Wegen  $r_1 1 = 1 \in [\varepsilon]$  ist  $\delta([\varepsilon], 1) = [\varepsilon]$ .
- 3 Wegen  $r_2 0 = 00 \notin [\varepsilon] \cup [0]$  ist  $r_3 = 00$  und  $\delta([0], 0) = [00]$ .
- 4 Wegen  $r_2 1 = 01 \notin [\varepsilon] \cup [0] \cup [00]$  ist  $r_4 = 01$  und  $\delta([0], 1) = [01]$ .
- 5 Wegen  $r_3 0 = 000 \in [00]$  ist  $\delta([00], 0) = [00]$ .
- 6 Wegen  $r_3 1 = 001 \in [01]$  ist  $\delta([00], 1) = [01]$ .
- 7 Wegen  $r_4 0 = 010 \in [0]$  ist  $\delta([01], 0) = [0]$ .
- 8 Wegen  $r_4 1 = 011 \in [\varepsilon]$

$r$	$\varepsilon$	0	00	01
$[r0]$	$[0]$	$[00]$	$[00]$	$[0]$
$[r1]$	$[\varepsilon]$	$[01]$	$[01]$	



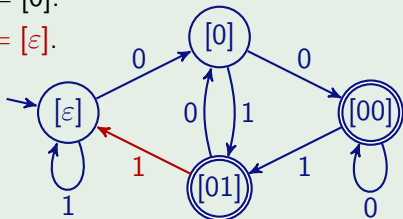
# Direkte Konstruktion des Äquivalenzklassen-DFA $M_{R_L}$ aus $L$

## Beispiel

Für die Sprache  $L = \{x_1 \cdots x_n \in \{0, 1\}^* \mid x_{n-1} = 0\}$  lässt sich  $M_{R_L}$  ausgehend von  $r_1 = \varepsilon$  wie folgt konstruieren:

- 1 Wegen  $r_1 0 = 0 \notin [\varepsilon]$  ist  $r_2 = 0$  und  $\delta([\varepsilon], 0) = [0]$ .
- 2 Wegen  $r_1 1 = 1 \in [\varepsilon]$  ist  $\delta([\varepsilon], 1) = [\varepsilon]$ .
- 3 Wegen  $r_2 0 = 00 \notin [\varepsilon] \cup [0]$  ist  $r_3 = 00$  und  $\delta([0], 0) = [00]$ .
- 4 Wegen  $r_2 1 = 01 \notin [\varepsilon] \cup [0] \cup [00]$  ist  $r_4 = 01$  und  $\delta([0], 1) = [01]$ .
- 5 Wegen  $r_3 0 = 000 \in [00]$  ist  $\delta([00], 0) = [00]$ .
- 6 Wegen  $r_3 1 = 001 \in [01]$  ist  $\delta([00], 1) = [01]$ .
- 7 Wegen  $r_4 0 = 010 \in [0]$  ist  $\delta([01], 0) = [0]$ .
- 8 Wegen  $r_4 1 = 011 \in [\varepsilon]$  ist  $\delta([01], 1) = [\varepsilon]$ .

$r$	$\varepsilon$	0	00	01
$[r0]$	$[0]$	$[00]$	$[00]$	$[0]$
$[r1]$	$[\varepsilon]$	$[01]$	$[01]$	$[\varepsilon]$



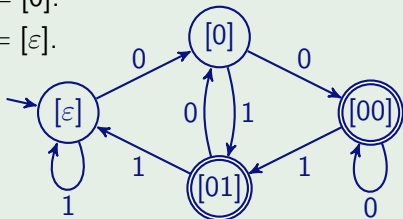
# Direkte Konstruktion des Äquivalenzklassen-DFA $M_{R_L}$ aus $L$

## Beispiel

Für die Sprache  $L = \{x_1 \cdots x_n \in \{0, 1\}^* \mid x_{n-1} = 0\}$  lässt sich  $M_{R_L}$  ausgehend von  $r_1 = \varepsilon$  wie folgt konstruieren:

- 1 Wegen  $r_1 0 = 0 \notin [\varepsilon]$  ist  $r_2 = 0$  und  $\delta([\varepsilon], 0) = [0]$ .
- 2 Wegen  $r_1 1 = 1 \in [\varepsilon]$  ist  $\delta([\varepsilon], 1) = [\varepsilon]$ .
- 3 Wegen  $r_2 0 = 00 \notin [\varepsilon] \cup [0]$  ist  $r_3 = 00$  und  $\delta([0], 0) = [00]$ .
- 4 Wegen  $r_2 1 = 01 \notin [\varepsilon] \cup [0] \cup [00]$  ist  $r_4 = 01$  und  $\delta([0], 1) = [01]$ .
- 5 Wegen  $r_3 0 = 000 \in [00]$  ist  $\delta([00], 0) = [00]$ .
- 6 Wegen  $r_3 1 = 001 \in [01]$  ist  $\delta([00], 1) = [01]$ .
- 7 Wegen  $r_4 0 = 010 \in [0]$  ist  $\delta([01], 0) = [0]$ .
- 8 Wegen  $r_4 1 = 011 \in [\varepsilon]$  ist  $\delta([01], 1) = [\varepsilon]$ .

$r$	$\varepsilon$	0	00	01
$[r0]$	$[0]$	$[00]$	$[00]$	$[0]$
$[r1]$	$[\varepsilon]$	$[01]$	$[01]$	$[\varepsilon]$



# Nachweis von $L \notin \text{REG}$ mittels Myhill und Nerode

## Satz

Die Sprache  $L = \{a^n b^n \mid n \geq 0\}$  ist nicht regulär.

## Beweis

Wegen

$$b^i \in L_{a^i} \Delta L_{a^j} \text{ für } i \neq j$$

hat  $R_L$  unendlichen Index. □

# Charakterisierungen der Klasse REG

## Korollar

Sei  $L$  eine Sprache. Dann sind folgende Aussagen äquivalent:

- $L$  ist regulär,
- es gibt einen DFA  $M$  mit  $L = L(M)$ ,
- es gibt einen NFA  $N$  mit  $L = L(N)$ ,
- es gibt einen regulären Ausdruck  $\gamma$  mit  $L = L(\gamma)$ ,
- die Äquivalenzrelation  $R_L$  hat endlichen Index.

## Ausblick

Wir werden nun noch eine weitere Charakterisierung von REG kennenlernen, nämlich durch reguläre Grammatiken.



# Erzeugung der regulären Ausdrücke mit einer Grammatik

Eine elegante Methode, Sprachen zu beschreiben, sind Grammatiken. Implizit haben wir hiervon bei der Definition der regulären Ausdrücke schon Gebrauch gemacht.

## Beispiel

Die Sprache  $RA$  aller regulären Ausdrücke über einem Alphabet  $\Sigma = \{a_1, \dots, a_k\}$  lässt sich aus dem Symbol  $R$  unter Anwendung folgender Regeln erzeugen:

$$R \rightarrow \emptyset,$$

$$R \rightarrow \epsilon,$$

$$R \rightarrow a_i, \quad i = 1, \dots, k,$$

$$R \rightarrow RR,$$

$$R \rightarrow (R|R),$$

$$R \rightarrow (R)^*.$$

# Definition einer Grammatik

## Definition

Eine **Grammatik** ist ein 4-Tupel  $G = (V, \Sigma, P, S)$ , wobei

- $V$  eine endliche Menge von **Variablen** (auch **Nichtterminalsymbole** genannt),
- $\Sigma$  das **Terminalalphabet**,
- $P \subseteq (V \cup \Sigma)^+ \times (V \cup \Sigma)^*$  eine endliche Menge von **Regeln** (oder **Produktionen**) und
- $S \in V$  die **Startvariable** ist.

## Bemerkung

Für  $(u, v) \in P$  schreiben wir auch kurz  $u \rightarrow_G v$  bzw.  $u \rightarrow v$ , wenn die benutzte Grammatik aus dem Kontext ersichtlich ist.

# Die von einer Grammatik erzeugte Sprache

- Ein Wort  $\beta \in (V \cup \Sigma)^*$  ist aus einem Wort  $\alpha \in (V \cup \Sigma)^+$  **in einem Schritt ableitbar** (kurz:  $\alpha \Rightarrow_G \beta$ ), falls eine Regel  $u \rightarrow_G v$  und Wörter  $l, r \in (V \cup \Sigma)^*$  existieren mit

$$\alpha = lur \text{ und } \beta = lvr.$$

Hierfür schreiben wir auch  $lur \Rightarrow_G lvr$ .

- Eine Folge  $\sigma = (l_0, u_0, r_0), \dots, (l_m, u_m, r_m)$  von Tripeln  $(l_i, u_i, r_i)$  heißt **Ableitung von  $\beta$  aus  $\alpha$** , falls gilt:
  - $l_0 u_0 r_0 = \alpha$ ,  $l_m u_m r_m = \beta$  und
  - $l_i \underline{u_i} r_i \Rightarrow l_{i+1} u_{i+1} r_{i+1}$  für  $i = 0, \dots, m-1$ .

Die **Länge** der Ableitung  $\sigma$  ist  $m$  und wir notieren  $\sigma$  auch in der Form

$$l_0 \underline{u_0} r_0 \Rightarrow l_1 \underline{u_1} r_1 \Rightarrow \dots \Rightarrow l_{m-1} \underline{u_{m-1}} r_{m-1} \Rightarrow l_m u_m r_m.$$

- Die durch  $G$  **erzeugte Sprache** ist  $L(G) = \{x \in \Sigma^* \mid S \Rightarrow_G^* x\}$ .
- Ein Wort  $\alpha \in (V \cup \Sigma)^*$  mit  $S \Rightarrow_G^* \alpha$  heißt **Satzform** von  $G$ .

# Ableitungen in einer Grammatik

## Zur Erinnerung:

- $\Rightarrow^*$  bezeichnet die reflexive, transitive Hülle der Relation  $\Rightarrow$ , d.h.  $\alpha \Rightarrow^* \beta$  bedeutet, dass es ein  $n \geq 0$  gibt mit  $\alpha \Rightarrow^n \beta$ .  
Hierzu sagen wir auch,  $\beta$  ist aus  $\alpha$  (in  $n$  Schritten) ableitbar.
- $\Rightarrow^n$  bezeichnet das  $n$ -fache Produkt der Relation  $\Rightarrow$ , d.h. es gilt  $\alpha \Rightarrow^n \beta$ , falls Wörter  $\alpha_0, \dots, \alpha_n$  existieren mit
  - $\alpha_0 = \alpha, \alpha_n = \beta$  und
  - $\alpha_i \Rightarrow \alpha_{i+1}$  für  $i = 0, \dots, n-1$ .

# Ableitung eines Wortes

## Beispiel

- Wir betrachten nochmals die Grammatik

$$G = (\{R\}, \Sigma \cup \{\emptyset, \epsilon, (, ), |, * \}, P, R)$$

für die Sprache aller regulären Ausdrücke über  $\Sigma$  mit den Regeln

$$P: \begin{aligned} R &\rightarrow \emptyset, \epsilon, a, a \in \Sigma \\ R &\rightarrow RR, (R|R), (R)^*. \end{aligned}$$

- Der reguläre Ausdruck  $(01)^*(\epsilon|\emptyset)$  über  $\Sigma = \{0, 1\}$  lässt sich in  $G$  aus dem Startsymbol  $R$  wie folgt ableiten:

$$\begin{aligned} \underline{R} &\Rightarrow \underline{RR} \Rightarrow (\underline{R})^* R \Rightarrow (\underline{RR})^* \underline{R} \Rightarrow (\underline{RR})^* (\underline{R|R}) \\ &\Rightarrow (\underline{0R})^* (\underline{R|R}) \Rightarrow (0\underline{1})^* (\underline{R|R}) \Rightarrow (01)^* (\underline{\epsilon|R}) \Rightarrow (01)^* (\underline{\epsilon|\emptyset}) \end{aligned}$$

# Die Chomsky-Hierarchie

Man unterscheidet vier Typen von Grammatiken  $G = (V, \Sigma, P, S)$ .

## Definition

- 1  $G$  heißt **vom Typ 3** oder **regulär**, falls für alle Regeln  $u \rightarrow v$  gilt:  
 $u \in V$  und  $v \in \Sigma V \cup \Sigma \cup \{\varepsilon\}$ ,  
(d.h. alle Regeln haben die Form  $A \rightarrow aB$ ,  $A \rightarrow a$  oder  $A \rightarrow \varepsilon$ ).
- 2  $G$  heißt **vom Typ 2** oder **kontextfrei**, falls für alle Regeln  $u \rightarrow v$  gilt:  
 $u \in V$ , (d.h. alle Regeln haben die Form  $A \rightarrow \alpha$ ).
- 3  $G$  heißt **vom Typ 1** oder **kontextsensitiv**, falls für alle Regeln  $u \rightarrow v$  gilt:  
 $|v| \geq |u|$ , (mit Ausnahme der  $\varepsilon$ -Sonderregel, s. unten).
- 4 Jede Grammatik ist automatisch **vom Typ 0**.

## Die $\varepsilon$ -Sonderregel

In einer kontextsensitiven Grammatik ist auch die Regel  $S \rightarrow \varepsilon$  zulässig.  
Aber nur, wenn das Startsymbol  $S$  in keiner Regel rechts vorkommt.

# Die Chomsky-Hierarchie

## Beispiel

- Wir betrachten nochmals die Grammatik

$$G = (\{R\}, \Sigma \cup \{\emptyset, \epsilon, (, ), |, * \}, P, R)$$

für die Sprache aller regulären Ausdrücke über  $\Sigma$  mit den Regeln

$$P : \begin{aligned} R &\rightarrow \emptyset, \epsilon, a, a \in \Sigma \\ R &\rightarrow RR, (R|R), (R)^*. \end{aligned}$$

- Da auf der linken Seite jeder Regel eine einzelne Variable steht, ist  $G$  kontextfrei.
- Offenbar ist  $G$  aber keine reguläre Grammatik, da zwar die  $\|\Sigma\| + 2$  Regeln  $R \rightarrow \emptyset, \epsilon, a, a \in \Sigma$ , die geforderte Form haben, nicht jedoch die drei Regeln  $R \rightarrow RR, (R|R), (R)^*$ .

# Die Chomsky-Hierarchie

- Eine Sprache heißt vom Typ  $i$  bzw. regulär, kontextfrei oder kontextsensitiv, falls sie von einer entsprechenden Grammatik erzeugt wird.

- Damit erhalten wir die neuen Sprachklassen

$$\text{CFL} = \{L(G) \mid G \text{ ist eine kontextfreie Grammatik}\}$$

und

*(context free languages)*

$$\text{CSL} = \{L(G) \mid G \text{ ist eine kontextsensitive Grammatik}\}$$

*(context sensitive languages).*

- Da die Klasse der Typ 0 Sprachen mit der Klasse der rekursiv aufzählbaren Sprachen übereinstimmt, bezeichnen wir diese Sprachklasse mit

$$\text{RE} = \{L(G) \mid G \text{ ist eine Grammatik}\}$$

*(recursively enumerable languages).*



# Die Chomsky-Hierarchie

- Wir werden bald beweisen, dass die Sprachklassen

$$\text{REG} \subset \text{CFL} \subset \text{CSL} \subset \text{RE}$$

eine Hierarchie bilden (d.h. die Inklusionen sind echt), die so genannte **Chomsky-Hierarchie**.

- Zunächst rechtfertigen wir jedoch die Bezeichnung **regulär** für die regulären Grammatiken und für die von ihnen erzeugten Sprachen.

# Reguläre Grammatiken

## Satz

$\text{REG} = \{L(G) \mid G \text{ ist eine reguläre Grammatik}\}.$

Beweis von  $\text{REG} \subseteq \{L(G) \mid G \text{ ist eine reguläre Grammatik}\}$

- Sei  $M = (Z, \Sigma, \delta, q_0, E)$  ein DFA.
- Wir konstruieren eine reguläre Grammatik  $G$  mit  $L(G) = L(M)$ .
- Betrachte die Grammatik  $G = (V, \Sigma, P, S)$  mit  $V = Z$ ,  $S = q_0$  und

$$P = \{q \rightarrow ap \mid \delta(q, a) = p\} \cup \{q \rightarrow \varepsilon \mid q \in E\}.$$

# Reguläre Grammatiken

## Beweis von $\text{REG} \subseteq \{L(G) \mid G \text{ ist eine reguläre Grammatik}\}$

- Betrachte die Grammatik  $G = (V, \Sigma, P, S)$  mit  $V = Z$ ,  $S = q_0$  und
$$P = \{q \rightarrow ap \mid \delta(q, a) = p\} \cup \{q \rightarrow \varepsilon \mid q \in E\}.$$

- Dann gilt für alle Wörter  $x = x_1 \cdots x_n \in \Sigma^*$ :

$$x \in L(M) \Leftrightarrow \exists q_1, \dots, q_{n-1} \in Z \exists q_n \in E :$$

$$\delta(q_{i-1}, x_i) = q_i \text{ für } i = 1, \dots, n$$

$$\Leftrightarrow \exists q_1, \dots, q_n \in V :$$

$$q_{i-1} \rightarrow x_i q_i \text{ für } i = 1, \dots, n \text{ und } q_n \rightarrow \varepsilon$$

$$\Leftrightarrow \exists q_1, \dots, q_n \in V :$$

$$q_0 \Rightarrow^i x_1 \cdots x_i q_i \text{ für } i = 1, \dots, n \text{ und } q_n \rightarrow \varepsilon$$

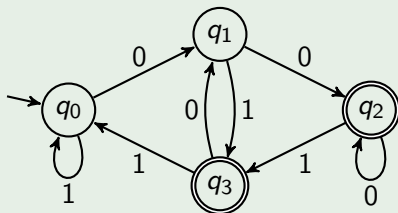
$$\Leftrightarrow x \in L(G)$$



# Reguläre Grammatiken

## Beispiel

Für den DFA



erhalten wir die Grammatik  $G = (\{q_0, q_1, q_2, q_3\}, \{0, 1\}, P, q_0)$  mit

$$P : q_0 \rightarrow 1q_0, 0q_1,$$

$$q_1 \rightarrow 0q_2, 1q_3,$$

$$q_2 \rightarrow 0q_2, 1q_3, \varepsilon,$$

$$q_3 \rightarrow 0q_1, 1q_0, \varepsilon.$$

# Reguläre Grammatiken

- Offensichtlich lässt sich obige Konstruktion einer Grammatik  $G$  aus einem DFA  $M$  umdrehen, falls  $G$  keine Regeln der Form  $A \rightarrow a$  enthält.
- Für den Beweis der Rückrichtung genügt es daher, alle Regeln dieser Form zu eliminieren.

## Lemma

Zu jeder regulären Grammatik  $G = (V, \Sigma, P, S)$  gibt es eine äquivalente reguläre Grammatik  $G'$ , die keine Regeln der Form  $A \rightarrow a$  hat.

## Beweis

Betrachte die Grammatik  $G' = (V', \Sigma, P', S)$  mit

$$V' = V \cup \{X_{neu}\} \text{ und}$$

$$P' = \{A \rightarrow aX_{neu} \mid A \rightarrow_G a\} \cup \{X_{neu} \rightarrow \varepsilon\} \cup P \setminus (V \times \Sigma).$$



# Reguläre Grammatiken

## Beispiel

- Betrachte die Grammatik  $G = (\{A, B, C\}, \{a, b\}, P, A)$  mit

$$P : A \rightarrow aB, bC, \varepsilon,$$

$$B \rightarrow aC, bA, b,$$

$$C \rightarrow aA, bB, a.$$

- Wir ersetzen die Regeln  $B \rightarrow b$  und  $C \rightarrow a$  durch die Regeln  $B \rightarrow bD$  und  $C \rightarrow aD$  und fügen die Regel  $D \rightarrow \varepsilon$  hinzu.
- Damit erhalten wir die Grammatik  $G' = (\{A, B, C, D\}, \{a, b\}, P', A)$  mit

$$P' : A \rightarrow aB, bC, \varepsilon,$$

$$B \rightarrow aC, bA, bD,$$

$$C \rightarrow aA, bB, aD,$$

$$D \rightarrow \varepsilon.$$

# Reguläre Grammatiken

## Beweis von $\{L(G) \mid G \text{ ist eine reguläre Grammatik}\} \subseteq \text{REG}$

- Sei  $G = (V, \Sigma, P, S)$  eine reguläre Grammatik, die keine Regeln der Form  $A \rightarrow a$  enthält.
- Drehen wir obige Konstruktion einer Grammatik aus einem DFA um, so erhalten wir den NFA

$$M = (Z, \Sigma, \delta, \{S\}, E)$$

mit  $Z = V$ ,  $\delta(A, a) = \{B \mid A \rightarrow_G aB\}$  und  $E = \{A \mid A \rightarrow_G \varepsilon\}$ .

- Genau wie oben folgt dann  $L(M) = L(G)$ . □

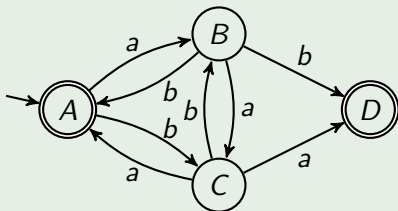
# Reguläre Grammatiken

## Beispiel (Fortsetzung)

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

$$\begin{aligned}P' : \quad & A \rightarrow aB, bC, \varepsilon, \\ & B \rightarrow aC, bA, bD, \\ & C \rightarrow aA, bB, aD, \\ & D \rightarrow \varepsilon.\end{aligned}$$

führt auf den NFA





# Charakterisierungen der Klasse REG

## Korollar

Sei  $L$  eine Sprache. Dann sind folgende Aussagen äquivalent:

- $L$  ist regulär,
- es gibt einen DFA  $M$  mit  $L = L(M)$ ,
- es gibt einen NFA  $N$  mit  $L = L(N)$ ,
- es gibt einen regulären Ausdruck  $\gamma$  mit  $L = L(\gamma)$ ,
- die Äquivalenzrelation  $R_L$  hat endlichen Index,
- es gibt eine reguläre Grammatik  $G$  mit  $L = L(G)$ .

# Das Pumping-Lemma

## Frage

Wie lässt sich möglichst einfach zeigen, dass eine Sprache nicht regulär ist?

## Antwort

Oft führt die Kontraposition folgender Aussage zum Ziel.

## Satz (Pumping-Lemma für reguläre Sprachen)

Zu jeder regulären Sprache  $L$  gibt es eine Zahl  $l \geq 0$ , so dass sich alle Wörter  $x \in L$  mit  $|x| \geq l$  in  $x = uvw$  zerlegen lassen mit

- 1  $v \neq \varepsilon$ ,
- 2  $|uv| \leq l$  und
- 3  $uv^i w \in L$  für alle  $i \geq 0$ .

Das kleinste solche  $l$  wird auch die **Pumping-Zahl** von  $L$  genannt.

# Das Pumping-Lemma

## Beispiel

- Die Sprache

$$L = \{x \in \{a, b\}^* \mid \#_a(x) - \#_b(x) \equiv_3 1\}$$

lässt sich „pumpen“ (mit Pumping-Zahl  $l = 3$ ).

- Sei  $x \in L$  beliebig mit  $|x| \geq 3$ .
  - 1. Fall:  $x$  hat das Präfix ab.  
Zerlege  $x = uvw$  mit  $u = \varepsilon$  und  $v = ab$ .
  - 2. Fall:  $x$  hat das Präfix aab.  
Zerlege  $x = uvw$  mit  $u = a$  und  $v = ab$ .
  - 3. Fall:  $x$  hat das Präfix aaa.  
Zerlege  $x = uvw$  mit  $u = \varepsilon$  und  $v = aaa$ .
  - Restliche Fälle (Präfixe ba, bba und bbb): analog.

# Das Pumping-Lemma

## Beispiel

- Eine **endliche** Sprache  $L$  lässt sich wie folgt „pumpen“.
- Sei

$$l = \begin{cases} 0, & L = \emptyset, \\ 1 + \max_{x \in L} |x|, & \text{sonst.} \end{cases}$$

- Dann lässt sich jedes Wort  $x \in L$  der Länge  $|x| \geq l$  „pumpen“ (da solche Wörter gar nicht existieren).
- Zudem gibt es im Fall  $l > 0$  ein Wort  $x \in L$  der Länge  $l - 1$ , das sich nicht „pumpen“ lässt.
- Also hat  $L$  die Pumping-Zahl  $l$ .

# Das Pumping-Lemma

## Satz (Pumping-Lemma für reguläre Sprachen)

Zu jeder regulären Sprache  $L$  gibt es eine Zahl  $l$ , so dass sich alle Wörter  $x \in L$  mit  $|x| \geq l$  in  $x = uvw$  zerlegen lassen mit

- 1  $v \neq \varepsilon$ ,
- 2  $|uv| \leq l$  und
- 3  $uv^i w \in L$  für alle  $i \geq 0$ .

Das kleinste solche  $l$  wird auch die **Pumping-Zahl** von  $L$  genannt.

# Das Pumping-Lemma

## Beweis

- Sei  $G = (V, \Sigma, P, S)$  eine reguläre Grammatik für  $L$ , die keine Regeln der Form  $A \rightarrow a$  enthält, und sei

$$\underline{A_0} \Rightarrow x_1 \underline{A_1} \Rightarrow x_1 x_2 \underline{A_2} \Rightarrow \cdots \Rightarrow x_1 x_2 \cdots x_n \underline{A_n} \Rightarrow x_1 x_2 \cdots x_n$$

eine beliebige Ableitung von  $x = x_1 \cdots x_n \in L$  aus  $A_0 = S$ .

- Ist  $|x| \geq l := \|V\|$ , so muss unter  $A_0, \dots, A_l$  eine Variable  $A$  mehrfach vorkommen, d.h. es ex.  $0 \leq j < k \leq l$  mit  $A_j = A_k = A$ .
- Somit können wir die Ableitung von  $x$  wie folgt zerlegen:

$$A_0 \Rightarrow^j x_1 \cdots x_j A_j = uA \Rightarrow^{k-j} ux_{j+1} \cdots x_k A_k = uvA \Rightarrow^{n+1-k} uvw,$$

wobei  $u = x_1 \cdots x_j$ ,  $v = x_{j+1} \cdots x_k$  und  $w = x_{k+1} \cdots x_n$  ist.

- Dann gilt  $|v| = k - j \geq 1$  (d.h.  $v \neq \varepsilon$ ),  $k = |uv| \leq l$  und für  $i \geq 0$  zeigt die Ableitung

$$A_0 \Rightarrow^j uA \Rightarrow^{(k-j)i} uv^i A \Rightarrow^{n+1-k} uv^i w,$$

dass  $uv^i w \in L$  ist.



# Das Pumping-Lemma

## Alternativbeweis über DFA.

- Sei  $M = (Z, \Sigma, \delta, q_0, E)$  ein DFA mit  $l$  Zuständen und sei  $x = x_1 \cdots x_n \in L$  mit  $n = |x| \geq l$ .
- Dann muss  $M(x)$  nach spätestens  $l$  Schritten einen Zustand zum zweiten Mal annehmen, d.h. es ex.  $0 \leq j < k \leq l$  und  $z \in Z$  mit

$$\hat{\delta}(q_0, x_1 \cdots x_j) = z \text{ und}$$

$$\hat{\delta}(q_0, x_1 \cdots x_j x_{j+1} \cdots x_k) = z.$$

- Setze  $u = x_1 \cdots x_j$ ,  $v = x_{j+1} \cdots x_k$  und  $w = x_{k+1} \cdots x_n$ .
- Dann gilt  $|v| = k - j \geq 1$  (d.h.  $v \neq \varepsilon$ ),  $k = |uv| \leq l$ .
- Zudem gehört für alle  $i \geq 0$  das Wort  $uv^i w$  zu  $L$ , da wegen  $\hat{\delta}(z, v) = z$

$$\hat{\delta}(q_0, uv^i w) = \hat{\delta}(\underbrace{\hat{\delta}(\hat{\delta}(q_0, u), v^i)}_z, w) = \hat{\delta}(\underbrace{\hat{\delta}(z, v^i)}_z, w) = \hat{\delta}(q_0, x)$$

in  $E$  ist.



# Kontraposition des Pumping-Lemmas

Um also  $L \notin \text{REG}$  zu zeigen, genügt es,

- für jede Zahl  $l \geq 0$  ein Wort  $x \in L$  der Länge  $|x| \geq l$  zu finden, so dass
- für jede Zerlegung  $x = uvw$  mindestens eine der folgenden drei Bedingungen verletzt ist:
  - ①  $v \neq \varepsilon$ ,
  - ②  $|uv| \leq l$  oder
  - ③  $uv^i w \in L$  für alle  $i \geq 0$ .

## Beispiel

Die Sprache

$$L = \{a^n b^n \mid n \geq 0\}$$

ist nicht regulär:

- Für jede Zahl  $l \geq 0$  enthält  $L$  das Wort  $x = a^l b^l$  mit  $|x| = 2l \geq l$ .
- Für jede Zerlegung  $x = uvw$  von  $x = a^l b^l$  mit
  - ①  $v \neq \varepsilon$ist die Bedingung
  - ③  $uv^i w \in L$für alle  $i \geq 2$  verletzt.



# Kontraposition des Pumping-Lemmas

Um also  $L \notin \text{REG}$  zu zeigen, genügt es,

- für jede Zahl  $l \geq 0$  ein Wort  $x \in L$  der Länge  $|x| \geq l$  zu finden, so dass
- für jede Zerlegung  $x = uvw$  mindestens eine der folgenden drei Bedingungen verletzt ist:
  - ①  $v \neq \varepsilon$ ,
  - ②  $|uv| \leq l$  oder
  - ③  $uv^i w \in L$  für alle  $i \geq 0$ .

Beispiel ( $L = \{a^{n^2} \mid n \geq 0\} \notin \text{REG}$ )

- Für jede Zahl  $l \geq 0$  enthält  $L$  ein Wort  $x$  mit  $|x| = l^2 \geq l$ .
- Für jede Zerlegung  $x = uvw$  mit  $|u| = r$ ,  $|v| = s$ ,  $|w| = t$  und
  - ①  $v \neq \varepsilon$  (d.h.  $s \geq 1$ ) sowie
  - ②  $|uv| \leq l$  (d.h.  $r + s \leq l$ )ist die Bedingung
  - ③  $uv^2w \in L$

verletzt, da  $r + 2s + t = l^2 + s$  keine Quadratzahl ist:

$$l^2 < l^2 + s < l^2 + l + 1 \leq (l + 1)^2.$$

# Kontraposition des Pumping-Lemmas

Um also  $L \notin \text{REG}$  zu zeigen, genügt es,

- für jede Zahl  $l \geq 0$  ein Wort  $x \in L$  der Länge  $|x| \geq l$  zu finden, so dass
- für jede Zerlegung  $x = uvw$  mindestens eine der folgenden drei Bedingungen verletzt ist:
  - ①  $v \neq \varepsilon$ ,
  - ②  $|uv| \leq l$  oder
  - ③  $uv^i w \in L$  für alle  $i \geq 0$ .

Beispiel ( $L = \{a^p \mid p \text{ prim}\} \notin \text{REG}$ )

- Für jede Zahl  $l \geq 0$  enthält  $L$  ein Wort  $x$  mit  $|x| = p \geq l$ .
- Für jede Zerlegung  $x = uvw$  mit  $|v| = s$  und
  - ①  $v \neq \varepsilon$  (d.h.  $s \geq 1$ )ist die Bedingung
  - ③  $uv^i w \in L$wegen
$$|uv^i w| = p + (i - 1)s$$
für  $i = p + 1$  verletzt, da dann
$$|uv^i w| = p + ps = p(s + 1)$$
ist.

# Grenzen des Pumping-Lemmas

## Bemerkung

- Mit dem Pumping-Lemma können nicht alle Sprachen  $L \notin \text{REG}$  als nicht regulär nachgewiesen werden, da seine Umkehrung falsch ist.

- Betrachte die Sprache

$$L = \{a^i b^j c^k \mid i = 0 \text{ oder } j = k\}.$$

- Da jedes Wort  $x \in L$  mit Ausnahme von  $\varepsilon$  „gepumpt“ werden kann, hat  $L$  die Pumping-Zahl 1.
- Allerdings ist  $L$  nicht regulär (siehe Übungen).

# Kontextfreie Sprachen

## Bemerkung

- Wie wir gesehen haben, ist folgende Sprache nicht regulär:

$$L = \{a^n b^n \mid n \geq 0\}.$$

- Es ist aber leicht, eine kontextfreie Grammatik für  $L$  zu finden:

$$G = (\{S\}, \{a, b\}, \{S \rightarrow aSb, S \rightarrow \varepsilon\}, S).$$

- Damit ist klar, dass die Klasse der regulären Sprachen echt in der Klasse der kontextfreien Sprachen enthalten ist:

$$\text{REG} \subsetneq \text{CFL}.$$

- Als nächstes wollen wir zeigen, dass die Klasse der kontextfreien Sprachen wiederum echt in der Klasse der kontextsensitiven Sprachen enthalten ist:

$$\text{CFL} \subsetneq \text{CSL}.$$

## Kontextfreie Sprachen sind auch kontextsensitiv

- Kontextfreie Grammatiken sind dadurch charakterisiert, dass sie nur Regeln der Form  $A \rightarrow \alpha$  haben.
- Dies lässt die Verwendung von beliebigen  $\varepsilon$ -Regeln der Form  $A \rightarrow \varepsilon$  zu.
- Eine kontextsensitive Grammatik darf dagegen höchstens die  $\varepsilon$ -Regel  $S \rightarrow \varepsilon$  haben.
- Voraussetzung hierfür ist, dass  $S$  das Startsymbol ist und dieses nicht auf der rechten Seite einer Regel vorkommt.
- Daher sind nicht alle kontextfreien Grammatiken kontextsensitiv.
- Wir werden jedoch sehen, dass sich zu jeder kontextfreien Grammatik eine äquivalente kontextsensitive Grammatik konstruieren lässt.

# Entfernen von $\varepsilon$ -Regeln

## Satz

Zu jeder kontextfreien Grammatik  $G = (V, \Sigma, P, S)$  gibt es eine kontextfreie Grammatik  $G' = (V, \Sigma, P', S)$  ohne  $\varepsilon$ -Regeln mit  $L(G') = L(G) \setminus \{\varepsilon\}$ .

## Beweis

- Zuerst berechnen wir die Menge  $E = \{A \in V \mid A \Rightarrow^* \varepsilon\}$  aller  *$\varepsilon$ -ableitbaren* Variablen:

```
1   $E' := \{A \in V \mid A \rightarrow \varepsilon\}$ 
2  repeat
3     $E := E'$ 
4     $E' := E \cup \{A \in V \mid \exists B_1, \dots, B_k \in E : A \rightarrow B_1 \cdots B_k\}$ 
5  until  $E = E'$ 
```

- Nun bilden wir  $P'$  wie folgt:

$\left\{ A \rightarrow \alpha' \mid \begin{array}{l} \text{es ex. eine Regel } A \rightarrow_G \alpha, \text{ so dass } \alpha' \neq \varepsilon \text{ aus } \alpha \text{ durch} \\ \text{Entfernen von beliebig vielen Variablen } A \in E \text{ entsteht} \end{array} \right\}$ .  $\square$

# Entfernen von $\epsilon$ -Regeln

## Beispiel

Betrachte die Grammatik  $G = (\{S, T, U, X, Y, Z\}, \{a, b, c\}, P, S)$  mit

$$P: \quad S \rightarrow aY, bX, Z; \quad Y \rightarrow bS, aYY; \quad T \rightarrow U; \\ X \rightarrow aS, bXX; \quad Z \rightarrow \epsilon, S, T, cZ; \quad U \rightarrow abc.$$

- Berechnung von  $E$ :

$E'$	$\{Z\}$	$\{Z, S\}$
$E$	$\{Z, S\}$	$\{Z, S\}$

- Entferne  $Z \rightarrow \epsilon$  und füge  $X \rightarrow a$  (wegen  $X \rightarrow aS$ ),  $Y \rightarrow b$  (wegen  $Y \rightarrow bS$ ) und  $Z \rightarrow c$  (wegen  $Z \rightarrow cZ$ ) hinzu:

$$P': \quad S \rightarrow aY, bX, Z; \quad Y \rightarrow b, bS, aYY; \quad T \rightarrow U; \\ X \rightarrow a, aS, bXX; \quad Z \rightarrow c, S, T, cZ; \quad U \rightarrow abc.$$

# Die Chomsky-Hierarchie

## Satz

Zu jeder kontextfreien Grammatik  $G = (V, \Sigma, P, S)$  gibt es eine kontextfreie Grammatik  $G' = (V, \Sigma, P', S)$  ohne  $\varepsilon$ -Regeln mit  $L(G') = L(G) \setminus \{\varepsilon\}$ .

## Korollar

$\text{REG} \subsetneq \text{CFL} \subseteq \text{CSL} \subseteq \text{RE}$ .

## Beweis

- Es ist nur noch die Inklusion  $\text{CFL} \subseteq \text{CSL}$  zu zeigen.
- Nach obigem Satz ex. zu  $L \in \text{CFL}$  eine kontextfreie Grammatik  $G = (V, \Sigma, P, S)$  ohne  $\varepsilon$ -Regeln mit  $L(G) = L \setminus \{\varepsilon\}$ .
- Da  $G$  dann auch kontextsensitiv ist, folgt hieraus im Fall  $\varepsilon \notin L$  unmittelbar  $L(G) = L \in \text{CSL}$ .
- Im Fall  $\varepsilon \in L$  erzeugt die kontextsensitive Grammatik

$$G' = (V \cup \{S'\}, \Sigma, P \cup \{S' \rightarrow S, \varepsilon\}, S')$$

die Sprache  $L(G') = L$ , d.h.  $L \in \text{CSL}$ . □



# Abschlusseigenschaften von CFL

## Satz

CFL ist abgeschlossen unter Vereinigung, Produkt und Sternhülle.

## Beweis

Seien  $G_1 = (V_1, \Sigma, P_1, S_1)$  und  $G_2 = (V_2, \Sigma, P_2, S_2)$  kontextfreie Grammatiken mit  $V_1 \cap V_2 = \emptyset$  und sei  $S$  eine neue Variable.

Dann erzeugen die kontextfreien Grammatiken

$$G_3 = (V_1 \cup V_2 \cup \{S\}, \Sigma, P_1 \cup P_2 \cup \{S \rightarrow S_1, S_2\}, S)$$

die Vereinigung  $L(G_3) = L(G_1) \cup L(G_2)$ ,

$$G_4 = (V_1 \cup V_2 \cup \{S\}, \Sigma, P_1 \cup P_2 \cup \{S \rightarrow S_1 S_2\}, S)$$

das Produkt  $L(G_4) = L(G_1)L(G_2)$  und

$$G_5 = (V_1 \cup \{S\}, \Sigma, P_1 \cup \{S \rightarrow S_1 S, \varepsilon\}, S)$$

die Sternhülle  $L(G_1)^*$ .



# Abschlusseigenschaften von CFL

## Satz

CFL ist abgeschlossen unter Vereinigung, Produkt und Sternhülle.

## Frage

Ist die Klasse CFL auch abgeschlossen unter

- Durchschnitt und
- Komplement?

## Antwort

Nein.

Hierzu müssen wir für bestimmte Sprachen nachweisen, dass sie nicht kontextfrei sind. Dies gelingt mit einem Pumping-Lemma für kontextfreie Sprachen, für dessen Beweis wir Grammatiken in Chomsky-Normalform benötigen.

# Chomsky-Normalform

## Definition

Eine Grammatik  $(V, \Sigma, P, S)$  ist in **Chomsky-Normalform (CNF)**, falls  $P \subseteq V \times (V^2 \cup \Sigma)$  ist, also alle Regeln die Form  $A \rightarrow BC$  oder  $A \rightarrow a$  haben.

Um eine kontextfreie Grammatik in Chomsky-Normalform zu bringen, müssen wir neben den  $\varepsilon$ -Regeln  $A \rightarrow \varepsilon$  auch sämtliche Variablenumbenennungen  $A \rightarrow B$  loswerden.

## Definition

Regeln der Form  $A \rightarrow B$  heißen **Variablenumbenennungen**.

# Entfernen von Variablenumbenennungen

## Satz

Zu jeder kontextfreien Grammatik  $G$  ex. eine kontextfreie Grammatik  $G'$  ohne Variablenumbenennungen mit  $L(G') = L(G)$ .

## Beweis

- Zuerst entfernen wir sukzessive alle Zyklen

$$A_1 \rightarrow A_2 \rightarrow \dots \rightarrow A_k \rightarrow A_1.$$

Hierzu entfernen wir diese Regeln aus  $P$  und ersetzen alle Vorkommen der Variablen  $A_2, \dots, A_k$  in den übrigen Regeln durch  $A_1$ .

(Sollte sich unter den entfernten Variablen  $A_2, \dots, A_k$  die Startvariable  $S$  befinden, so sei  $A_1$  die neue Startvariable.)

# Entfernen von Variablenumbenennungen

## Beispiel (Fortsetzung)

$P: S \rightarrow aY, bX, Z; \quad Y \rightarrow b, bS, aYY; \quad T \rightarrow U;$   
 $X \rightarrow a, aS, bXX; \quad Z \rightarrow c, S, T, cZ; \quad U \rightarrow abc.$

- Entferne den Zyklus  $S \rightarrow Z \rightarrow S$  und ersetze alle Vorkommen von  $Z$  durch  $S$ :

$S \rightarrow aY, bX, c, T, cS; \quad Y \rightarrow b, bS, aYY; \quad T \rightarrow U;$   
 $X \rightarrow a, aS, bXX; \quad U \rightarrow abc.$

# Entfernen von Variablenumbenennungen

## Satz

Zu jeder kontextfreien Grammatik  $G$  ex. eine kontextfreie Grammatik  $G'$  ohne Variablenumbenennungen mit  $L(G') = L(G)$ .

## Beweis

- Zuerst entfernen wir alle Zyklen

$$A_1 \rightarrow A_2 \rightarrow \cdots \rightarrow A_k \rightarrow A_1.$$

- Nun werden wir sukzessive die restlichen Variablenumbenennungen los, indem wir
  - eine Regel  $A \rightarrow B$  wählen, so dass in  $P$  keine Variablenumbenennung  $B \rightarrow C$  mit  $B$  auf der linken Seite existiert,
  - diese Regel  $A \rightarrow B$  aus  $P$  entfernen und
  - für jede Regel  $B \rightarrow \alpha$  in  $P$  die Regel  $A \rightarrow \alpha$  zu  $P$  hinzunehmen.  $\square$

# Entfernen von Variablenumbenennungen

## Beispiel (Fortsetzung)

$$S \rightarrow aY, bX, c, T, cS; \quad Y \rightarrow b, bS, aYY; \quad T \rightarrow U;$$
$$X \rightarrow a, aS, bXX; \quad U \rightarrow abc.$$

- Entferne die Regel  $T \rightarrow U$  und füge die Regel  $T \rightarrow abc$  hinzu (wegen  $U \rightarrow abc$ ):

$$S \rightarrow aY, bX, c, T, cS; \quad Y \rightarrow b, bS, aYY; \quad T \rightarrow abc;$$
$$X \rightarrow a, aS, bXX; \quad U \rightarrow abc.$$

- Entferne dann auch die Regel  $S \rightarrow T$  und füge die Regel  $S \rightarrow abc$  (wegen  $T \rightarrow abc$ ) hinzu:

$$S \rightarrow abc, aY, bX, c, cS; \quad Y \rightarrow b, bS, aYY; \quad T \rightarrow abc;$$
$$X \rightarrow a, aS, bXX; \quad U \rightarrow abc.$$

- Da  $T$  und  $U$  nirgends mehr auf der rechten Seite vorkommen, können wir die Regeln  $T \rightarrow abc$  und  $U \rightarrow abc$  weglassen:

$$S \rightarrow abc, aY, bX, c, cS; \quad Y \rightarrow b, bS, aYY; \quad X \rightarrow a, aS, bXX.$$

# Chomsky-Normalform

## Definition

Eine Grammatik  $(V, \Sigma, P, S)$  ist in **Chomsky-Normalform (CNF)**, falls  $P \subseteq V \times (V^2 \cup \Sigma)$  ist, also alle Regeln die Form  $A \rightarrow BC$  oder  $A \rightarrow a$  haben.

## Anwendungen der Chomsky-Normalform

- CNF-Grammatiken bilden die Basis für eine effiziente Lösung des Wortproblems für kontextfreie Sprachen.
- Zudem ermöglichen sie den Beweis des Pumping-Lemmas für kontextfreie Sprachen.



# Entfernen von $\varepsilon$ -Regeln und von Variablenumbenennungen

Bereits gezeigt:

## Korollar

Zu jeder kontextfreien Grammatik  $G$  ex. eine kontextfreie Grammatik  $G'$  ohne  $\varepsilon$ -Regeln und ohne Variablenumbenennungen mit  $L(G') = L(G) \setminus \{\varepsilon\}$ .

Noch zu zeigen:

## Satz

Zu jeder kontextfreien Sprache  $L \in \text{CFL}$  gibt es eine CNF-Grammatik  $G'$  mit  $L(G') = L \setminus \{\varepsilon\}$ .

# Umwandlung in Chomsky-Normalform

## Satz

Zu jeder kontextfreien Sprache  $L \in \text{CFL}$  gibt es eine CNF-Grammatik  $G'$  mit  $L(G') = L \setminus \{\varepsilon\}$ .

## Beweis

- Sei  $G = (V, \Sigma, P, S)$  eine kontextfreie Grammatik ohne  $\varepsilon$ -Regeln und ohne Variablenumbenennungen für  $L \setminus \{\varepsilon\}$ .
- Wir transformieren  $G$  wie folgt in eine CNF-Grammatik.
- Füge für jedes Terminalsymbol  $a \in \Sigma$  eine neue Variable  $X_a$  zu  $V$  und eine neue Regel  $X_a \rightarrow a$  zu  $P$  hinzu.
- Ersetze alle Vorkommen von  $a$  durch  $X_a$ , außer wenn  $a$  alleine auf der rechten Seite einer Regel steht.
- Ersetze jede Regel  $A \rightarrow B_1 \cdots B_k$ ,  $k \geq 3$ , durch die  $k - 1$  Regeln
$$A \rightarrow B_1 A_1, A_1 \rightarrow B_2 A_2, \dots, A_{k-3} \rightarrow B_{k-2} A_{k-2}, A_{k-2} \rightarrow B_{k-1} B_k,$$
wobei  $A_1, \dots, A_{k-2}$  neue Variablen sind. □

# Umwandlung in Chomsky-Normalform

## Beispiel (Fortsetzung)

- Betrachte die Regeln

$$P: S \rightarrow abc, aY, bX, cS, c; \quad X \rightarrow aS, bXX, a; \quad Y \rightarrow bS, aYY, b.$$

- Ersetze  $a$ ,  $b$  und  $c$  durch  $A$ ,  $B$  und  $C$  (außer wenn sie alleine vorkommen) und füge die Regeln  $A \rightarrow a$ ,  $B \rightarrow b$ ,  $C \rightarrow c$  hinzu:

$$S \rightarrow ABC, AY, BX, CS, c; \quad X \rightarrow AS, BXX, a;$$
$$Y \rightarrow BS, AYY, b; \quad A \rightarrow a; \quad B \rightarrow b; \quad C \rightarrow c.$$

- Ersetze die Regeln  $S \rightarrow ABC$ ,  $X \rightarrow BXX$  und  $Y \rightarrow AYY$  durch die Regeln  $S \rightarrow AS'$ ,  $S' \rightarrow BC$ ,  $X \rightarrow BX'$ ,  $X' \rightarrow XX$  und  $Y \rightarrow AY'$ ,  $Y' \rightarrow YY$ :

$$S \rightarrow AS', AY, BX, CS, c; \quad S' \rightarrow BC; \quad X \rightarrow AS, BX', a; \quad X' \rightarrow XX;$$
$$Y \rightarrow BS, AY', b; \quad Y' \rightarrow YY; \quad A \rightarrow a; \quad B \rightarrow b; \quad C \rightarrow c.$$

# Syntaxbäume

## Definition

Wir ordnen einer Ableitung

$$\underline{A_0} \Rightarrow l_1 \underline{A_1} r_1 \Rightarrow \cdots \Rightarrow l_{m-1} \underline{A_{m-1}} r_{m-1} \Rightarrow \alpha_m$$

den **Syntaxbaum** (oder **Ableitungsbaum**, engl. *parse tree*)  $T_m$  zu, wobei die Bäume  $T_0, \dots, T_m$  induktiv wie folgt definiert sind:

- $T_0$  besteht aus einem einzigen Knoten, der mit  $A_0$  markiert ist.
- Wird im  $(i+1)$ -ten Ableitungsschritt die Regel  $A_i \rightarrow v_1 \cdots v_k$  mit  $v_1, \dots, v_k \in \Sigma \cup V$  angewandt, so entsteht  $T_{i+1}$  aus  $T_i$ , indem wir das Blatt  $A_i$  durch folgenden Unterbaum ersetzen:

$$\begin{array}{ccc} k > 0 : & A_i & k = 0 : A_i \\ & \begin{array}{c} / \quad \backslash \\ v_1 \cdots v_k \end{array} & \begin{array}{c} | \\ \varepsilon \end{array} \end{array}$$

- Hierbei stellen wir uns die Kanten von oben nach unten gerichtet und die Kinder  $v_1 \cdots v_k$  von links nach rechts geordnet vor.

# Syntaxbäume

## Beispiel

- Betrachte die Grammatik  $G = (\{S\}, \{a, b, c\}, \{S \rightarrow aSbS, \varepsilon\}, S)$

# Syntaxbäume

## Beispiel

- Betrachte die Grammatik  $G = (\{S\}, \{a, b, c\}, \{S \rightarrow aSbS, \varepsilon\}, S)$  und die Ableitung

$S$

# Syntaxbäume

## Beispiel

- Betrachte die Grammatik  $G = (\{S\}, \{a, b, c\}, \{S \rightarrow aSbS, \varepsilon\}, S)$  und die Ableitung

$$\underline{S} \Rightarrow aSbS$$

# Syntaxbäume

## Beispiel

- Betrachte die Grammatik  $G = (\{S\}, \{a, b, c\}, \{S \rightarrow aSbS, \varepsilon\}, S)$  und die Ableitung

$$\underline{S} \Rightarrow a\underline{S}bS \Rightarrow aa\underline{S}bSbS$$



# Syntaxbäume

## Beispiel

- Betrachte die Grammatik  $G = (\{S\}, \{a, b, c\}, \{S \rightarrow aSbS, \varepsilon\}, S)$  und die Ableitung

$$\underline{S} \Rightarrow a\underline{S}bS \Rightarrow aaSb\underline{S}bS \Rightarrow aaSbbS$$

# Syntaxbäume

## Beispiel

- Betrachte die Grammatik  $G = (\{S\}, \{a, b, c\}, \{S \rightarrow aSbS, \varepsilon\}, S)$  und die Ableitung

$$\underline{S} \Rightarrow a\underline{S}bS \Rightarrow aaSb\underline{S}bS \Rightarrow aa\underline{S}bbS \Rightarrow aabbS$$

# Syntaxbäume

## Beispiel

- Betrachte die Grammatik  $G = (\{S\}, \{a, b, c\}, \{S \rightarrow aSbS, \varepsilon\}, S)$  und die Ableitung

$$\underline{S} \Rightarrow a\underline{S}bS \Rightarrow aaSb\underline{S}bS \Rightarrow aa\underline{S}bbS \Rightarrow aabb\underline{S} \Rightarrow aabb$$

# Syntaxbäume

## Beispiel

- Betrachte die Grammatik  $G = (\{S\}, \{a, b, c\}, \{S \rightarrow aSbS, \varepsilon\}, S)$  und die Ableitung

$$\underline{S} \Rightarrow a\underline{S}bS \Rightarrow aaSb\underline{S}bS \Rightarrow aa\underline{S}bbS \Rightarrow aabb\underline{S} \Rightarrow aabb$$

- Die zugehörigen Syntaxbäume sind dann

# Syntaxbäume

## Beispiel

- Betrachte die Grammatik  $G = (\{S\}, \{a, b, c\}, \{S \rightarrow aSbS, \varepsilon\}, S)$  und die Ableitung

$$\underline{S} \Rightarrow a\underline{S}bS \Rightarrow aaSb\underline{S}bS \Rightarrow aa\underline{S}bbS \Rightarrow aabb\underline{S} \Rightarrow aabb$$

- Die zugehörigen Syntaxbäume sind dann

$$T_0: S$$

S

# Syntaxbäume

## Beispiel

- Betrachte die Grammatik  $G = (\{S\}, \{a, b, c\}, \{S \rightarrow aSbS, \varepsilon\}, S)$  und die Ableitung

$$\underline{S} \Rightarrow a\underline{S}bS \Rightarrow aaSb\underline{S}bS \Rightarrow aa\underline{S}bbS \Rightarrow aabb\underline{S} \Rightarrow aabb$$

- Die zugehörigen Syntaxbäume sind dann

$$T_1: \begin{array}{c} S \\ / \quad | \quad \backslash \\ a \quad S \quad b \quad S \end{array}$$

$$\underline{S} \Rightarrow aSbS$$

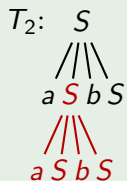
# Syntaxbäume

## Beispiel

- Betrachte die Grammatik  $G = (\{S\}, \{a, b, c\}, \{S \rightarrow aSbS, \varepsilon\}, S)$  und die Ableitung

$$\underline{S} \Rightarrow a\underline{S}bS \Rightarrow aaSb\underline{S}bS \Rightarrow aa\underline{S}bbS \Rightarrow aabb\underline{S} \Rightarrow aabb$$

- Die zugehörigen Syntaxbäume sind dann



$$\underline{S} \Rightarrow a\underline{S}bS \Rightarrow aa\underline{S}bSbS$$

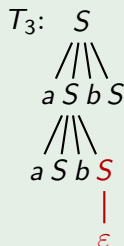
# Syntaxbäume

## Beispiel

- Betrachte die Grammatik  $G = (\{S\}, \{a, b, c\}, \{S \rightarrow aSbS, \varepsilon\}, S)$  und die Ableitung

$$\underline{S} \Rightarrow a\underline{S}bS \Rightarrow aaSb\underline{S}bS \Rightarrow aa\underline{S}bbS \Rightarrow aabb\underline{S} \Rightarrow aabb$$

- Die zugehörigen Syntaxbäume sind dann



$$\underline{S} \Rightarrow a\underline{S}bS \Rightarrow aaSb\underline{S}bS \Rightarrow aaSbbS$$



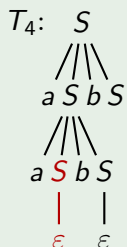
# Syntaxbäume

## Beispiel

- Betrachte die Grammatik  $G = (\{S\}, \{a, b, c\}, \{S \rightarrow aSbS, \varepsilon\}, S)$  und die Ableitung

$$\underline{S} \Rightarrow a\underline{S}bS \Rightarrow aaSb\underline{S}bS \Rightarrow aa\underline{S}bbS \Rightarrow aabb\underline{S} \Rightarrow aabb$$

- Die zugehörigen Syntaxbäume sind dann



$$\underline{S} \Rightarrow a\underline{S}bS \Rightarrow aaSb\underline{S}bS \Rightarrow aa\underline{S}bbS \Rightarrow aabbS$$

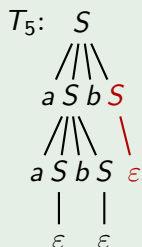
# Syntaxbäume

## Beispiel

- Betrachte die Grammatik  $G = (\{S\}, \{a, b, c\}, \{S \rightarrow aSbS, \varepsilon\}, S)$  und die Ableitung

$$\underline{S} \Rightarrow a\underline{S}bS \Rightarrow aaSb\underline{S}bS \Rightarrow aa\underline{S}bbS \Rightarrow aabb\underline{S} \Rightarrow aabb$$

- Die zugehörigen Syntaxbäume sind dann



$$\underline{S} \Rightarrow a\underline{S}bS \Rightarrow aaSb\underline{S}bS \Rightarrow aa\underline{S}bbS \Rightarrow aabb\underline{S} \Rightarrow aabb$$

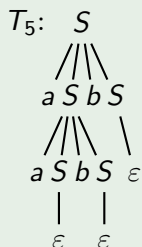
# Syntaxbäume

## Beispiel

- Betrachte die Grammatik  $G = (\{S\}, \{a, b, c\}, \{S \rightarrow aSbS, \varepsilon\}, S)$  und die Ableitung

$$\underline{S} \Rightarrow a\underline{S}bS \Rightarrow aaSb\underline{S}bS \Rightarrow aa\underline{S}bbS \Rightarrow aabb\underline{S} \Rightarrow aabb$$

- Die zugehörigen Syntaxbäume sind dann



$$\underline{S} \Rightarrow a\underline{S}bS \Rightarrow aaSb\underline{S}bS \Rightarrow aa\underline{S}bbS \Rightarrow aabb\underline{S} \Rightarrow aabb$$

# Links- und Rechtsableitungen

## Definition

Sei  $G = (V, \Sigma, P, S)$  eine kontextfreie Grammatik.

- Eine Ableitung

$$\underline{S} \Rightarrow l_1 \underline{A_1} r_1 \Rightarrow \cdots \Rightarrow l_{m-1} \underline{A_{m-1}} r_{m-1} \Rightarrow \alpha_m$$

heißt **Linksableitung** von  $\alpha_m$  (kurz  $S \Rightarrow_L^* \alpha_m$ ), falls in jedem Ableitungsschritt die am weitesten links stehende Variable ersetzt wird, d.h. es gilt  $l_i \in \Sigma^*$  für  $i = 1, \dots, m - 1$ .

- **Rechtsableitungen**  $S_0 \Rightarrow_R^* \alpha_m$  sind analog definiert.
- $G$  heißt **mehrdeutig**, wenn es ein Wort  $x \in L(G)$  gibt, das zwei verschiedene Linksableitungen hat.

## Leicht zu sehen:

Für alle  $x \in \Sigma^*$  gilt:  $x \in L(G) \Leftrightarrow S \Rightarrow^* x \Leftrightarrow S \Rightarrow_L^* x \Leftrightarrow S \Rightarrow_R^* x$ .

# Syntaxbäume

## Beispiel

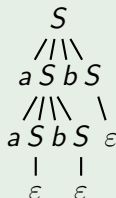
- Die Grammatik  $G = (\{S\}, \{a, b, c\}, \{S \rightarrow aSbS, \varepsilon\}, S)$  ist eindeutig.
- Dies liegt daran, dass in keiner Satzform von  $G$  die Variable  $S$  von einem  $a$  gefolgt wird.
- Daher muss jede Linksableitung eines Wortes  $x \in L(G)$  die am weitesten links stehende Variable der aktuellen Satzform  $\alpha S \beta$  genau dann nach  $aSbS$  expandieren, falls das Präfix  $\alpha$  in  $x$  von einem  $a$  gefolgt wird.
- Dagegen ist die Grammatik  $G = (\{S\}, \{a, b, c\}, \{S \rightarrow aSbS, ab, \varepsilon\}, S)$  mehrdeutig, da das Wort  $x = ab$  2 verschiedene Linksableitungen hat:

$$\underline{S} \Rightarrow ab \text{ und } \underline{S} \Rightarrow a\underline{S}bS \Rightarrow ab\underline{S} \Rightarrow ab.$$

# Syntaxbäume

## Beispiel

- In  $G = (\{S\}, \{a, b, c\}, \{S \rightarrow aSbS, \varepsilon\}, S)$  führen insgesamt 8 Ableitungen auf den Syntaxbaum



$\underline{S} \Rightarrow a\underline{S}bS \Rightarrow aaSb\underline{S}bS \Rightarrow aa\underline{S}bbS \Rightarrow aabb\underline{S} \Rightarrow aabb$

$\underline{S} \Rightarrow a\underline{S}bS \Rightarrow aaSb\underline{S}bS \Rightarrow aaSbb\underline{S} \Rightarrow aa\underline{S}bb \Rightarrow aabb$

$\underline{S} \Rightarrow a\underline{S}bS \Rightarrow aa\underline{S}bSbS \Rightarrow aab\underline{S}bS \Rightarrow aabb\underline{S} \Rightarrow aabb$

$\underline{S} \Rightarrow a\underline{S}bS \Rightarrow aa\underline{S}bSbS \Rightarrow aabSb\underline{S} \Rightarrow aab\underline{S}b \Rightarrow aabb$

$\underline{S} \Rightarrow a\underline{S}bS \Rightarrow aaSbSb\underline{S} \Rightarrow aa\underline{S}bSb \Rightarrow aab\underline{S}b \Rightarrow aabb$

$\underline{S} \Rightarrow a\underline{S}bS \Rightarrow aaSbSb\underline{S} \Rightarrow aaSb\underline{S}b \Rightarrow aa\underline{S}bb \Rightarrow aabb$

$\underline{S} \Rightarrow aSb\underline{S} \Rightarrow a\underline{S}b \Rightarrow aa\underline{S}bSb \Rightarrow aab\underline{S}b \Rightarrow aabb$

$\underline{S} \Rightarrow aSb\underline{S} \Rightarrow a\underline{S}b \Rightarrow aaSb\underline{S}b \Rightarrow aa\underline{S}bb \Rightarrow aabb$

# Syntaxbäume und Linksableitungen

- Seien  $T_0, \dots, T_m$  die zu einer Ableitung  $S = \alpha_0 \Rightarrow \dots \Rightarrow \alpha_m$  gehörigen Syntaxbäume.
- Die Satzform  $\alpha_j$  ergibt sich aus  $T_j$ , indem wir die Blätter von  $T_j$  von links nach rechts zu einem Wort zusammensetzen.
- Auf den Syntaxbaum  $T_m$  führen neben  $\alpha_0 \Rightarrow \dots \Rightarrow \alpha_m$  alle Ableitungen, die sich von dieser nur in der Reihenfolge der Regelanwendungen unterscheiden.
- Dazu gehört genau eine Linksableitung.
- Linksableitungen und Syntaxbäume entsprechen sich also eineindeutig.
- Dasselbe gilt für Rechtsableitungen.
- Ist  $T$  Syntaxbaum einer CNF-Grammatik, so hat jeder Knoten in  $T$  höchstens zwei Kinder (d.h.  $T$  ist ein Binärbaum).

# Das Pumping-Lemma für kontextfreie Sprachen

Als erste Anwendung der Chomsky-Normalform beweisen wir das Pumping-Lemma für kontextfreie Sprachen.

## Satz (Pumping-Lemma für kontextfreie Sprachen)

Zu jeder kontextfreien Sprache  $L$  gibt es eine Zahl  $l$ , so dass sich alle Wörter  $z \in L$  mit  $|z| \geq l$  in  $z = uvwxy$  zerlegen lassen mit

- 1  $vx \neq \varepsilon$ ,
- 2  $|vwx| \leq l$  und
- 3  $uv^iwx^iy \in L$  für alle  $i \geq 0$ .

## Beispiel

- Betrachte die Sprache  $L = \{a^n b^n \mid n \geq 0\}$ .
- Dann lässt sich jedes Wort  $z = a^n b^n = a^{n-1} a b b^{n-1}$  in  $L$  mit  $|z| \geq 2$  pumpen:
  - Zerlege  $z$  in  $z = uvwxy$  mit  $u = a^{n-1}$ ,  $v = a$ ,  $w = \varepsilon$ ,  $x = b$ ,  $y = b^{n-1}$ .
  - Dann ist für alle  $i \geq 0$  das Wort  $uv^iwx^iy = a^{n-1} a^i b^i b^{n-1} \in L$ . ◀



# Abschätzung der Blätterzahl bei Binärbäumen

## Definition

Die **Tiefe** eines Baumes mit Wurzel  $w$  ist die maximale Pfadlänge von  $w$  zu einem Blatt.

## Lemma

Ein Binärbaum  $B$  der Tiefe  $\leq k$  hat  $\leq 2^k$  Blätter.

## Beweis durch Induktion über $k$ :

$k = 0$ : Ein Baum der Tiefe 0 kann nur einen Knoten haben.

$k \rightsquigarrow k + 1$ : Sei  $B$  ein Binärbaum der Tiefe  $\leq k + 1$ .

Dann hängen an  $B$ 's Wurzel maximal zwei Unterbäume.

Da deren Tiefe  $\leq k$  ist, haben sie nach IV  $\leq 2^k$  Blätter.

Also hat  $B \leq 2^{k+1}$  Blätter. □

# Mindesttiefe von Binärbäumen

## Lemma

Ein Binärbaum  $B$  der Tiefe  $\leq k$  hat  $\leq 2^k$  Blätter.

## Korollar

Ein Binärbaum  $B$  mit mehr als  $2^{k-1}$  Blättern hat mindestens die Tiefe  $k$ .

## Beweis

Würde ein Binärbaum  $B$  der Tiefe  $\leq k - 1$  mehr als  $2^{k-1}$  Blätter besitzen, so stünde dies im Widerspruch zu obigem Lemma.  $\square$

# Beweis des Pumping-Lemmas

## Satz (Pumping-Lemma für kontextfreie Sprachen)

Zu jeder kontextfreien Sprache  $L \in \text{CFL}$  gibt es eine Zahl  $l$ , so dass sich alle Wörter  $z \in L$  mit  $|z| \geq l$  in  $z = uvwxy$  zerlegen lassen mit

- 1  $vx \neq \varepsilon$ ,
- 2  $|vwx| \leq l$  und
- 3  $uv^iwx^iy \in L$  für alle  $i \geq 0$ .

## Beweis

- Sei  $G = (V, \Sigma, P, S)$  eine CNF-Grammatik für  $L \setminus \{\varepsilon\}$ .
- Ist nun  $z = z_1 \cdots z_n \in L$  mit  $n \geq 1$ , so ex. in  $G$  eine Ableitung

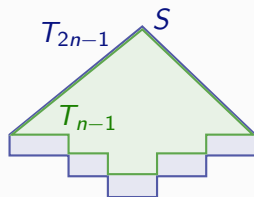
$$S = \alpha_0 \Rightarrow \alpha_1 \cdots \Rightarrow \alpha_m = z.$$

- Da  $G$  in CNF ist, werden hierbei genau  $n - 1$  Regeln der Form  $A \rightarrow BC$  und genau  $n$  Regeln der Form  $A \rightarrow a$  angewandt.

# Beweis des Pumping-Lemmas

## Beweis

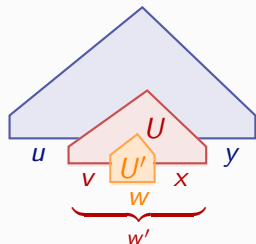
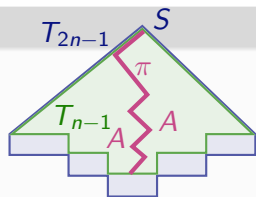
- Sei  $G = (V, \Sigma, P, S)$  eine CNF-Grammatik für  $L \setminus \{\varepsilon\}$ .
- Ist nun  $z = z_1 \cdots z_n \in L$  mit  $n \geq 1$ , so ex. in  $G$  eine Ableitung  
$$S = \alpha_0 \Rightarrow \alpha_1 \cdots \Rightarrow \alpha_m = z.$$
- Da  $G$  in CNF ist, werden hierbei genau  $n - 1$  Regeln der Form  $A \rightarrow BC$  und genau  $n$  Regeln der Form  $A \rightarrow a$  angewandt.
- Folglich ist  $m = 2n - 1$  und  $z$  hat den Syntaxbaum  $T_{2n-1}$ .
- Wir können annehmen, dass die  $n - 1$  Regeln der Form  $A \rightarrow BC$  vor den  $n$  Regeln der Form  $A \rightarrow a$  zur Anwendung kommen.
- Dann besteht  $\alpha_{n-1}$  aus  $n$  Variablen und  $T_{n-1}$  hat wie  $T_{2n-1}$  genau  $n$  Blätter.
- Setzen wir  $l = 2^k$ , wobei  $k = \|V\|$  ist, so hat  $T_{n-1}$  im Fall  $n \geq l$  mindestens  $l = 2^k > 2^{k-1}$  Blätter und daher mindestens die Tiefe  $k$ .



# Beweis des Pumping-Lemmas

## Beweis (Fortsetzung)

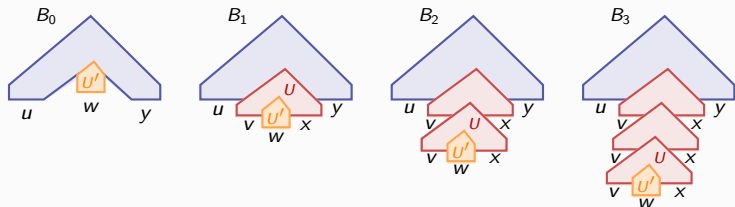
- Setzen wir  $l = 2^k$ , wobei  $k = \|V\|$  ist, so hat  $T_{n-1}$  im Fall  $n \geq l$  mindestens  $l = 2^k > 2^{k-1}$  Blätter und daher mindestens die Tiefe  $k$ .
- Sei  $\pi$  ein von der Wurzel ausgehender Pfad maximaler Länge in  $T_{n-1}$ .
- Dann hat  $\pi$  mindestens die Länge  $k$  und unter den letzten  $k + 1$  Knoten von  $\pi$  müssen zwei mit derselben Variablen  $A$  markiert sein.
- Seien  $U$  und  $U'$  die von diesen Knoten ausgehenden Unterbäume des vollständigen Syntaxbaums  $T_{2n-1}$ .
- Nun zerlegen wir  $z$  wie folgt:
  - $w'$  ist das Teilwort von  $z = uw'y$ , das von  $U$  erzeugt wird und
  - $w$  ist das Teilwort von  $w' = vwx$ , das von  $U'$  erzeugt wird.



# Beweis des Pumping-Lemmas

## Beweis (Schluss)

- Da  $U$  mehr Blätter hat als  $U'$ , ist  $vx \neq \varepsilon$  (Bedingung 1).
- Da der Baum  $U^* = U \cap T_{n-1}$  höchstens die Tiefe  $k$  hat (andernfalls wäre  $\pi$  nicht maximal), hat  $U^*$  (und damit  $U$ ) höchstens  $2^k = l$  Blätter.
- Folglich ist  $|vwx| \leq l$  (Bedingung 2).
- Schließlich lassen sich Syntaxbäume  $B_i$  für die Wörter  $uv^iwx^iy$ ,  $i \geq 0$ , wie folgt konstruieren (Bedingung 3):
  - $B_0$  entsteht aus  $B_1 = T_{2n-1}$ , indem wir  $U$  durch  $U'$  ersetzen.
  - $B_{i+1}$  entsteht aus  $B_i$ , indem wir  $U'$  durch  $U$  ersetzen:



# Anwendung des Pumping-Lemmas

## Beispiel

- Die Sprache  $\{a^n b^n c^n \mid n \geq 0\}$  ist nicht kontextfrei.
- Für eine vorgegebene Zahl  $l \geq 0$  hat nämlich  $z = a^l b^l c^l$  die Länge  $|z| = 3l \geq l$ .
- Dieses Wort lässt sich aber nicht pumpen:

Für jede Zerlegung  $z = uvwxy$  mit  $vx \neq \varepsilon$  und  $|vwx| \leq l$  gehört  $z' = uv^2wx^2y$  nicht zu  $L$ :

- Wegen  $vx \neq \varepsilon$  ist  $|z| < |z'|$ .
- Wegen  $|vwx| \leq l$  kann in  $vx$  nicht jedes der drei Zeichen  $a, b, c$  vorkommen.
- Kommt aber in  $vx$  beispielsweise kein  $a$  vor, so ist

$$\#_a(z') = \#_a(z) = l = |z|/3 < |z'|/3.$$

- Also kann  $z'$  nicht zu  $L$  gehören.

# Abschlusseigenschaften von CFL

Wie wir gesehen haben, ist die Klasse CFL abgeschlossen unter

- Vereinigung,
- Produkt und
- Sternhülle.

## Satz

CFL ist nicht abgeschlossen unter

- Durchschnitt und
- Komplement.



## Abschlusseigenschaften von CFL

Beweis von  $L_1, L_2 \in \text{CFL} \not\Rightarrow L_1 \cap L_2 \in \text{CFL}$

- Die beiden Sprachen

$$L_1 = \{a^n b^m c^m \mid n, m \geq 0\} \quad \text{und} \quad L_2 = \{a^n b^n c^m \mid n, m \geq 0\}$$

sind kontextfrei (siehe Übungen).

- Nicht jedoch ihr Schnitt  $L_1 \cap L_2 = \{a^n b^n c^n \mid n \geq 0\}$ .
- Also ist CFL nicht unter Durchschnitt abgeschlossen.

Beweis von  $L \in \text{CFL} \not\Rightarrow \bar{L} \in \text{CFL}$

Da CFL zwar unter Vereinigung aber nicht unter Schnitt abgeschlossen ist, kann CFL wegen de Morgan nicht unter Komplement abgeschlossen sein.

# Das Wortproblem für CFL

## Das Wortproblem für kontextfreie Grammatiken

Gegeben: Eine kontextfreie Grammatik  $G$  und ein Wort  $x$ .

Gefragt: Ist  $x \in L(G)$ ?

## Frage

Ist das Wortproblem für kontextfreie Grammatiken effizient entscheidbar?

# Der CYK-Algorithmus

## Satz

Das Wortproblem für kontextfreie Grammatiken ist effizient entscheidbar.

## Beweis

- Sei eine Grammatik  $G = (V, \Sigma, P, S)$  und ein Wort  $x = x_1 \cdots x_n$  gegeben.
- Falls  $x = \varepsilon$  ist, können wir effizient prüfen, ob  $S \Rightarrow^* \varepsilon$  gilt.
- Hierzu genügt es, die Menge  $E = \{A \in V \mid A \Rightarrow^* \varepsilon\}$  aller  $\varepsilon$ -ableitbaren Variablen zu berechnen und zu prüfen, ob  $S \in E$  ist.
- Andernfalls bringen wir  $G$  in CNF und starten den nach seinen Autoren **Cocke**, **Younger** und **Kasami** benannten **CYK-Algorithmus**.
- Dieser bestimmt mittels **dynamischer Programmierung** für  $l = 1, \dots, n$  und  $k = 1, \dots, n - l + 1$  die Menge  $V_{l,k}$  aller Variablen, aus denen das Teilwort  $x_k \cdots x_{k+l-1}$  ableitbar ist.
- Dann gilt  $x \in L(G) \Leftrightarrow S \in V_{n,1}$ .

# Berechnung der Mengen $V_{l,k}$

## Beweis (Schluss)

- Sei  $G = (V, \Sigma, P, S)$  eine CNF-Grammatik und sei  $x \in \Sigma^+$ .
- Dann lassen sich die Mengen

$$V_{l,k} = \{A \in V \mid A \Rightarrow^* x_k \cdots x_{k+l-1}\}$$

wie folgt bestimmen.

- Für  $l = 1$  gehört  $A$  zu  $V_{1,k}$ , falls die Regel  $A \rightarrow x_k$  existiert,

$$V_{1,k} = \{A \in V \mid A \rightarrow x_k\}.$$

- Für  $l > 1$  gehört  $A$  zu  $V_{l,k}$ , falls eine Zahl  $l' \in \{1, \dots, l-1\}$  und eine Regel  $A \rightarrow BC$  ex., so dass  $B \in V_{l',k}$  und  $C \in V_{l-l',k+l'}$  sind,

$$V_{l,k} = \{A \in V \mid \exists l' < l \exists B \in V_{l',k} \exists C \in V_{l-l',k+l'} : A \rightarrow BC\}.$$

- Jede Menge  $V_{l,k}$  lässt sich in Zeit  $O(l|G|) = O(n|G|)$  bestimmen.
- Da insgesamt  $n(n+1)/2 = O(n^2)$  Mengen  $V_{l,k}$  zu bestimmen sind, ist dies in Zeit  $O(n^3|G|)$  möglich. □

# Der CYK-Algorithmus

## Algorithmus CYK( $G, x$ )

```
1  Input: CNF-Grammatik  $G = (V, \Sigma, P, S)$  und Wort  $x = x_1 \cdots x_n$ 
2  for  $k := 1$  to  $n$  do
3       $V_{1,k} := \{A \in V \mid A \rightarrow x_k \in P\}$ 
4  for  $l := 2$  to  $n$  do
5      for  $k := 1$  to  $n - l + 1$  do
6           $V_{l,k} := \emptyset$ 
7          for  $l' := 1$  to  $l - 1$  do
8              for all  $A \rightarrow BC \in P$  do
9                  if  $B \in V_{l',k}$  and  $C \in V_{l-l',k+l'}$  then
10                      $V_{l,k} := V_{l,k} \cup \{A\}$ 
11  if  $S \in V_{n,1}$  then accept else reject
```

Der CYK-Algorithmus lässt sich leicht dahingehend modifizieren, dass er im Fall  $x \in L(G)$  auch einen Syntaxbaum  $T$  von  $x$  bestimmt.

# Der CYK-Algorithmus

## Beispiel

- Betrachte die CNF-Grammatik mit den Regeln

# Der CYK-Algorithmus

## Beispiel

- Betrachte die CNF-Grammatik mit den Regeln

$$P: S \rightarrow AS', AY, BX, CS, c, \quad S' \rightarrow BC, \quad X \rightarrow AS, BX', a, \quad X' \rightarrow XX, \\ Y \rightarrow BS, AY', b, \quad Y' \rightarrow YY, \quad A \rightarrow a, \quad B \rightarrow b, \quad C \rightarrow c.$$

# Der CYK-Algorithmus

## Beispiel

- Betrachte die CNF-Grammatik mit den Regeln

$$P: S \rightarrow AS', AY, BX, CS, c, \quad S' \rightarrow BC, \quad X \rightarrow AS, BX', a, \quad X' \rightarrow XX, \\ Y \rightarrow BS, AY', b, \quad Y' \rightarrow YY, \quad A \rightarrow a, \quad B \rightarrow b, \quad C \rightarrow c.$$

- Dann erhalten wir für das Wort  $x = abb$  folgende Mengen  $V_{l,k}$ :



# Der CYK-Algorithmus

## Beispiel

- Betrachte die CNF-Grammatik mit den Regeln

$P: S \rightarrow AS', AY, BX, CS, c, \quad S' \rightarrow BC, \quad X \rightarrow AS, BX', a, \quad X' \rightarrow XX,$   
 $Y \rightarrow BS, AY', b, \quad Y' \rightarrow YY, \quad A \rightarrow a, \quad B \rightarrow b, \quad C \rightarrow c.$

- Dann erhalten wir für das Wort  $x = abb$  folgende Mengen  $V_{l,k}$ :

$k:$	1	2	3			
	<table border="1"><tr><td><math>a</math></td></tr></table>	$a$	<table border="1"><tr><td><math>b</math></td></tr></table>	$b$	<table border="1"><tr><td><math>b</math></td></tr></table>	$b$
$a$						
$b$						
$b$						
$l: 1$	<table border="1"><tr><td></td></tr></table>		<table border="1"><tr><td></td></tr></table>		<table border="1"><tr><td></td></tr></table>	
2	<table border="1"><tr><td></td></tr></table>		<table border="1"><tr><td></td></tr></table>			
3	<table border="1"><tr><td></td></tr></table>					

# Der CYK-Algorithmus

## Beispiel

- Betrachte die CNF-Grammatik mit den Regeln

$P: S \rightarrow AS', AY, BX, CS, c, \quad S' \rightarrow BC, \quad X \rightarrow AS, BX', a, \quad X' \rightarrow XX,$   
 $Y \rightarrow BS, AY', b, \quad Y' \rightarrow YY, \quad A \rightarrow a, \quad B \rightarrow b, \quad C \rightarrow c.$

- Dann erhalten wir für das Wort  $x = abb$  folgende Mengen  $V_{l,k}$ :

$k:$	1	2	3			
	<table border="1"><tr><td><math>a</math></td></tr></table>	$a$	<table border="1"><tr><td><math>b</math></td></tr></table>	$b$	<table border="1"><tr><td><math>b</math></td></tr></table>	$b$
$a$						
$b$						
$b$						
$l: 1$	<table border="1"><tr><td><math>\{X, A\}</math></td></tr></table>	$\{X, A\}$	<table border="1"><tr><td></td></tr></table>		<table border="1"><tr><td></td></tr></table>	
$\{X, A\}$						
2	<table border="1"><tr><td></td></tr></table>		<table border="1"><tr><td></td></tr></table>			
3	<table border="1"><tr><td></td></tr></table>					

# Der CYK-Algorithmus

## Beispiel

- Betrachte die CNF-Grammatik mit den Regeln

$P: S \rightarrow AS', AY, BX, CS, c, \quad S' \rightarrow BC, \quad X \rightarrow AS, BX', a, \quad X' \rightarrow XX,$   
 $Y \rightarrow BS, AY', b, \quad Y' \rightarrow YY, \quad A \rightarrow a, \quad B \rightarrow b, \quad C \rightarrow c.$

- Dann erhalten wir für das Wort  $x = abb$  folgende Mengen  $V_{l,k}$ :

$k:$	1	2	3			
	<table border="1"><tr><td><math>a</math></td></tr></table>	$a$	<table border="1"><tr><td><math>b</math></td></tr></table>	$b$	<table border="1"><tr><td><math>b</math></td></tr></table>	$b$
$a$						
$b$						
$b$						
$l: 1$	<table border="1"><tr><td><math>\{X, A\}</math></td></tr></table>	$\{X, A\}$	<table border="1"><tr><td><math>\{Y, B\}</math></td></tr></table>	$\{Y, B\}$	<table border="1"><tr><td></td></tr></table>	
$\{X, A\}$						
$\{Y, B\}$						
2	<table border="1"><tr><td></td></tr></table>		<table border="1"><tr><td></td></tr></table>			
3	<table border="1"><tr><td></td></tr></table>					

# Der CYK-Algorithmus

## Beispiel

- Betrachte die CNF-Grammatik mit den Regeln

$P: S \rightarrow AS', AY, BX, CS, c, \quad S' \rightarrow BC, \quad X \rightarrow AS, BX', a, \quad X' \rightarrow XX,$   
 $Y \rightarrow BS, AY', b, \quad Y' \rightarrow YY, \quad A \rightarrow a, \quad B \rightarrow b, \quad C \rightarrow c.$

- Dann erhalten wir für das Wort  $x = abb$  folgende Mengen  $V_{l,k}$ :

$k:$	1	2	3			
	<table border="1"><tr><td><math>a</math></td></tr></table>	$a$	<table border="1"><tr><td><math>b</math></td></tr></table>	$b$	<table border="1"><tr><td><math>b</math></td></tr></table>	$b$
$a$						
$b$						
$b$						
$l: 1$	<table border="1"><tr><td><math>\{X, A\}</math></td></tr></table>	$\{X, A\}$	<table border="1"><tr><td><math>\{Y, B\}</math></td></tr></table>	$\{Y, B\}$	<table border="1"><tr><td><math>\{Y, B\}</math></td></tr></table>	$\{Y, B\}$
$\{X, A\}$						
$\{Y, B\}$						
$\{Y, B\}$						
2	<table border="1"><tr><td></td></tr></table>		<table border="1"><tr><td></td></tr></table>			
3	<table border="1"><tr><td></td></tr></table>					

# Der CYK-Algorithmus

## Beispiel

- Betrachte die CNF-Grammatik mit den Regeln

$P: S \rightarrow AS', AY, BX, CS, c, \quad S' \rightarrow BC, \quad X \rightarrow AS, BX', a, \quad X' \rightarrow XX,$   
 $Y \rightarrow BS, AY', b, \quad Y' \rightarrow YY, \quad A \rightarrow a, \quad B \rightarrow b, \quad C \rightarrow c.$

- Dann erhalten wir für das Wort  $x = abb$  folgende Mengen  $V_{l,k}$ :

$k:$	1	2	3			
	<table border="1"><tr><td><math>a</math></td></tr></table>	$a$	<table border="1"><tr><td><math>b</math></td></tr></table>	$b$	<table border="1"><tr><td><math>b</math></td></tr></table>	$b$
$a$						
$b$						
$b$						
$l: 1$	<table border="1"><tr><td><math>\{X, A\}</math></td></tr></table>	$\{X, A\}$	<table border="1"><tr><td><math>\{Y, B\}</math></td></tr></table>	$\{Y, B\}$	<table border="1"><tr><td><math>\{Y, B\}</math></td></tr></table>	$\{Y, B\}$
$\{X, A\}$						
$\{Y, B\}$						
$\{Y, B\}$						
2	<table border="1"><tr><td><math>\{S\}</math></td></tr></table>	$\{S\}$	<table border="1"><tr><td></td></tr></table>			
$\{S\}$						
3	<table border="1"><tr><td></td></tr></table>					

# Der CYK-Algorithmus

## Beispiel

- Betrachte die CNF-Grammatik mit den Regeln

$P: S \rightarrow AS', AY, BX, CS, c, \quad S' \rightarrow BC, \quad X \rightarrow AS, BX', a, \quad X' \rightarrow XX,$   
 $Y \rightarrow BS, AY', b, \quad Y' \rightarrow YY, \quad A \rightarrow a, \quad B \rightarrow b, \quad C \rightarrow c.$

- Dann erhalten wir für das Wort  $x = abb$  folgende Mengen  $V_{l,k}$ :

$k:$	1	2	3			
	<table border="1"><tr><td><math>a</math></td></tr></table>	$a$	<table border="1"><tr><td><math>b</math></td></tr></table>	$b$	<table border="1"><tr><td><math>b</math></td></tr></table>	$b$
$a$						
$b$						
$b$						
$l: 1$	<table border="1"><tr><td><math>\{X, A\}</math></td></tr></table>	$\{X, A\}$	<table border="1"><tr><td><math>\{Y, B\}</math></td></tr></table>	$\{Y, B\}$	<table border="1"><tr><td><math>\{Y, B\}</math></td></tr></table>	$\{Y, B\}$
$\{X, A\}$						
$\{Y, B\}$						
$\{Y, B\}$						
2	<table border="1"><tr><td><math>\{S\}</math></td></tr></table>	$\{S\}$	<table border="1"><tr><td><math>\{Y'\}</math></td></tr></table>	$\{Y'\}$		
$\{S\}$						
$\{Y'\}$						
3	<table border="1"><tr><td></td></tr></table>					

# Der CYK-Algorithmus

## Beispiel

- Betrachte die CNF-Grammatik mit den Regeln

$P: S \rightarrow AS', AY, BX, CS, c, \quad S' \rightarrow BC, \quad X \rightarrow AS, BX', a, \quad X' \rightarrow XX,$   
 $Y \rightarrow BS, AY', b, \quad Y' \rightarrow YY, \quad A \rightarrow a, \quad B \rightarrow b, \quad C \rightarrow c.$

- Dann erhalten wir für das Wort  $x = abb$  folgende Mengen  $V_{l,k}$ :

$k:$	1	2	3			
	<table border="1"><tr><td><math>a</math></td></tr></table>	$a$	<table border="1"><tr><td><math>b</math></td></tr></table>	$b$	<table border="1"><tr><td><math>b</math></td></tr></table>	$b$
$a$						
$b$						
$b$						
$l: 1$	<table border="1"><tr><td><math>\{X, A\}</math></td></tr></table>	$\{X, A\}$	<table border="1"><tr><td><math>\{Y, B\}</math></td></tr></table>	$\{Y, B\}$	<table border="1"><tr><td><math>\{Y, B\}</math></td></tr></table>	$\{Y, B\}$
$\{X, A\}$						
$\{Y, B\}$						
$\{Y, B\}$						
2	<table border="1"><tr><td><math>\{S\}</math></td></tr></table>	$\{S\}$	<table border="1"><tr><td><math>\{Y'\}</math></td></tr></table>	$\{Y'\}$		
$\{S\}$						
$\{Y'\}$						
3	<table border="1"><tr><td><math>\{Y\}</math></td></tr></table>	$\{Y\}$				
$\{Y\}$						

# Der CYK-Algorithmus

## Beispiel

- Betrachte die CNF-Grammatik mit den Regeln

$$P: S \rightarrow AS', AY, BX, CS, c, \quad S' \rightarrow BC, \quad X \rightarrow AS, BX', a, \quad X' \rightarrow XX, \\ Y \rightarrow BS, AY', b, \quad Y' \rightarrow YY, \quad A \rightarrow a, \quad B \rightarrow b, \quad C \rightarrow c.$$

- Dann erhalten wir für das Wort  $x = abb$  folgende Mengen  $V_{l,k}$ :

$k:$	1	2	3			
	<table border="1"><tr><td><math>a</math></td></tr></table>	$a$	<table border="1"><tr><td><math>b</math></td></tr></table>	$b$	<table border="1"><tr><td><math>b</math></td></tr></table>	$b$
$a$						
$b$						
$b$						
$l: 1$	<table border="1"><tr><td><math>\{X, A\}</math></td></tr></table>	$\{X, A\}$	<table border="1"><tr><td><math>\{Y, B\}</math></td></tr></table>	$\{Y, B\}$	<table border="1"><tr><td><math>\{Y, B\}</math></td></tr></table>	$\{Y, B\}$
$\{X, A\}$						
$\{Y, B\}$						
$\{Y, B\}$						
2	<table border="1"><tr><td><math>\{S\}</math></td></tr></table>	$\{S\}$	<table border="1"><tr><td><math>\{Y'\}</math></td></tr></table>	$\{Y'\}$		
$\{S\}$						
$\{Y'\}$						
3	<table border="1"><tr><td><math>\{Y\}</math></td></tr></table>	$\{Y\}$				
$\{Y\}$						

- Wegen  $S \notin V_{3,1}$  ist  $x \notin L(G)$ .



# Der CYK-Algorithmus

## Beispiel (Fortsetzung)

- Betrachte die CNF-Grammatik mit den Regeln

$P: S \rightarrow AS', AY, BX, CS, c, \quad S' \rightarrow BC, \quad X \rightarrow AS, BX', a, \quad X' \rightarrow XX,$   
 $Y \rightarrow BS, AY', b, \quad Y' \rightarrow YY, \quad A \rightarrow a, \quad B \rightarrow b, \quad C \rightarrow c.$

# Der CYK-Algorithmus

## Beispiel (Fortsetzung)

- Betrachte die CNF-Grammatik mit den Regeln

$P: S \rightarrow AS', AY, BX, CS, c, \quad S' \rightarrow BC, \quad X \rightarrow AS, BX', a, \quad X' \rightarrow XX,$   
 $Y \rightarrow BS, AY', b, \quad Y' \rightarrow YY, \quad A \rightarrow a, \quad B \rightarrow b, \quad C \rightarrow c.$

- Dagegen gehört das Wort  $y = aababb$  zu  $L(G)$ :

<i>a</i>	<i>a</i>	<i>b</i>	<i>a</i>	<i>b</i>	<i>b</i>

# Der CYK-Algorithmus

## Beispiel (Fortsetzung)

- Betrachte die CNF-Grammatik mit den Regeln

$P: S \rightarrow AS', AY, BX, CS, c, \quad S' \rightarrow BC, \quad X \rightarrow AS, BX', a, \quad X' \rightarrow XX,$   
 $Y \rightarrow BS, AY', b, \quad Y' \rightarrow YY, \quad A \rightarrow a, \quad B \rightarrow b, \quad C \rightarrow c.$

- Dagegen gehört das Wort  $y = aababb$  zu  $L(G)$ :

<i>a</i>	<i>a</i>	<i>b</i>	<i>a</i>	<i>b</i>	<i>b</i>
{X, A}	{X, A}	{Y, B}	{X, A}	{Y, B}	{Y, B}

# Der CYK-Algorithmus

## Beispiel (Fortsetzung)

- Betrachte die CNF-Grammatik mit den Regeln

$P: S \rightarrow AS', AY, BX, CS, c, \quad S' \rightarrow BC, \quad X \rightarrow AS, BX', a, \quad X' \rightarrow XX,$   
 $Y \rightarrow BS, AY', b, \quad Y' \rightarrow YY, \quad A \rightarrow a, \quad B \rightarrow b, \quad C \rightarrow c.$

- Dagegen gehört das Wort  $y = aababb$  zu  $L(G)$ :

	<i>a</i>	<i>a</i>	<i>b</i>	<i>a</i>	<i>b</i>	<i>b</i>
	{ <i>X</i> , <i>A</i> }	{ <i>X</i> , <i>A</i> }	{ <i>Y</i> , <i>B</i> }	{ <i>X</i> , <i>A</i> }	{ <i>Y</i> , <i>B</i> }	{ <i>Y</i> , <i>B</i> }
	{ <i>X'</i> }					

# Der CYK-Algorithmus

## Beispiel (Fortsetzung)

- Betrachte die CNF-Grammatik mit den Regeln

$P: S \rightarrow AS', AY, BX, CS, c, S' \rightarrow BC, X \rightarrow AS, BX', a, X' \rightarrow XX,$   
 $Y \rightarrow BS, AY', b, Y' \rightarrow YY, A \rightarrow a, B \rightarrow b, C \rightarrow c.$

- Dagegen gehört das Wort  $y = aababb$  zu  $L(G)$ :

	<i>a</i>	<i>a</i>	<i>b</i>	<i>a</i>	<i>b</i>	<i>b</i>
	{X, A}	{X, A}	{Y, B}	{X, A}	{Y, B}	{Y, B}
	{X'}	{S}				

# Der CYK-Algorithmus

## Beispiel (Fortsetzung)

- Betrachte die CNF-Grammatik mit den Regeln

$P: S \rightarrow AS', AY, BX, CS, c, S' \rightarrow BC, X \rightarrow AS, BX', a, X' \rightarrow XX,$   
 $Y \rightarrow BS, AY', b, Y' \rightarrow YY, A \rightarrow a, B \rightarrow b, C \rightarrow c.$

- Dagegen gehört das Wort  $y = aababb$  zu  $L(G)$ :

	<i>a</i>	<i>a</i>	<i>b</i>	<i>a</i>	<i>b</i>	<i>b</i>
	{X, A}	{X, A}	{Y, B}	{X, A}	{Y, B}	{Y, B}
	{X'}	{S}	{S}			

# Der CYK-Algorithmus

## Beispiel (Fortsetzung)

- Betrachte die CNF-Grammatik mit den Regeln

$P: S \rightarrow AS', AY, BX, CS, c, S' \rightarrow BC, X \rightarrow AS, BX', a, X' \rightarrow XX,$   
 $Y \rightarrow BS, AY', b, Y' \rightarrow YY, A \rightarrow a, B \rightarrow b, C \rightarrow c.$

- Dagegen gehört das Wort  $y = aababb$  zu  $L(G)$ :

	<i>a</i>	<i>a</i>	<i>b</i>	<i>a</i>	<i>b</i>	<i>b</i>
	{X, A}	{X, A}	{Y, B}	{X, A}	{Y, B}	{Y, B}
	{X'}	{S}	{S}	{S}		

# Der CYK-Algorithmus

## Beispiel (Fortsetzung)

- Betrachte die CNF-Grammatik mit den Regeln

$P: S \rightarrow AS', AY, BX, CS, c, \quad S' \rightarrow BC, \quad X \rightarrow AS, BX', a, \quad X' \rightarrow XX,$   
 $Y \rightarrow BS, AY', b, \quad Y' \rightarrow YY, \quad A \rightarrow a, \quad B \rightarrow b, \quad C \rightarrow c.$

- Dagegen gehört das Wort  $y = aababb$  zu  $L(G)$ :

	<i>a</i>	<i>a</i>	<i>b</i>	<i>a</i>	<i>b</i>	<i>b</i>
	{X, A}	{X, A}	{Y, B}	{X, A}	{Y, B}	{Y, B}
	{X'}	{S}	{S}	{S}	{Y'}	



# Der CYK-Algorithmus

## Beispiel (Fortsetzung)

- Betrachte die CNF-Grammatik mit den Regeln

$P: S \rightarrow AS', AY, BX, CS, c, \quad S' \rightarrow BC, \quad X \rightarrow AS, BX', a, \quad X' \rightarrow XX,$   
 $Y \rightarrow BS, AY', b, \quad Y' \rightarrow YY, \quad A \rightarrow a, \quad B \rightarrow b, \quad C \rightarrow c.$

- Dagegen gehört das Wort  $y = aababb$  zu  $L(G)$ :

$a$	$a$	$b$	$a$	$b$	$b$
$\{X, A\}$	$\{X, A\}$	$\{Y, B\}$	$\{X, A\}$	$\{Y, B\}$	$\{Y, B\}$
$\{X'\}$	$\{S\}$	$\{S\}$	$\{S\}$	$\{Y'\}$	
$\{X\}$					

# Der CYK-Algorithmus

## Beispiel (Fortsetzung)

- Betrachte die CNF-Grammatik mit den Regeln

$P: S \rightarrow AS', AY, BX, CS, c, \quad S' \rightarrow BC, \quad X \rightarrow AS, BX', a, \quad X' \rightarrow XX,$   
 $Y \rightarrow BS, AY', b, \quad Y' \rightarrow YY, \quad A \rightarrow a, \quad B \rightarrow b, \quad C \rightarrow c.$

- Dagegen gehört das Wort  $y = aababb$  zu  $L(G)$ :

$a$	$a$	$b$	$a$	$b$	$b$
$\{X, A\}$	$\{X, A\}$	$\{Y, B\}$	$\{X, A\}$	$\{Y, B\}$	$\{Y, B\}$
$\{X'\}$	$\{S\}$	$\{S\}$	$\{S\}$	$\{Y'\}$	
$\{X\}$	$\{X\}$				

# Der CYK-Algorithmus

## Beispiel (Fortsetzung)

- Betrachte die CNF-Grammatik mit den Regeln

$P: S \rightarrow AS', AY, BX, CS, c, S' \rightarrow BC, X \rightarrow AS, BX', a, X' \rightarrow XX,$   
 $Y \rightarrow BS, AY', b, Y' \rightarrow YY, A \rightarrow a, B \rightarrow b, C \rightarrow c.$

- Dagegen gehört das Wort  $y = aababb$  zu  $L(G)$ :

	<i>a</i>	<i>a</i>	<i>b</i>	<i>a</i>	<i>b</i>	<i>b</i>
	{X, A}	{X, A}	{Y, B}	{X, A}	{Y, B}	{Y, B}
	{X'}	{S}	{S}	{S}	{Y'}	
	{X}	{X}	{Y}			

# Der CYK-Algorithmus

## Beispiel (Fortsetzung)

- Betrachte die CNF-Grammatik mit den Regeln

$P: S \rightarrow AS', AY, BX, CS, c, S' \rightarrow BC, X \rightarrow AS, BX', a, X' \rightarrow XX,$   
 $Y \rightarrow BS, AY', b, Y' \rightarrow YY, A \rightarrow a, B \rightarrow b, C \rightarrow c.$

- Dagegen gehört das Wort  $y = aababb$  zu  $L(G)$ :

	<i>a</i>	<i>a</i>	<i>b</i>	<i>a</i>	<i>b</i>	<i>b</i>
	{X, A}	{X, A}	{Y, B}	{X, A}	{Y, B}	{Y, B}
	{X'}	{S}	{S}	{S}	{Y'}	
	{X}	{X}	{Y}	{Y}		

# Der CYK-Algorithmus

## Beispiel (Fortsetzung)

- Betrachte die CNF-Grammatik mit den Regeln

$P: S \rightarrow AS', AY, BX, CS, c, \quad S' \rightarrow BC, \quad X \rightarrow AS, BX', a, \quad X' \rightarrow XX,$   
 $Y \rightarrow BS, AY', b, \quad Y' \rightarrow YY, \quad A \rightarrow a, \quad B \rightarrow b, \quad C \rightarrow c.$

- Dagegen gehört das Wort  $y = aababb$  zu  $L(G)$ :

	<i>a</i>	<i>a</i>	<i>b</i>	<i>a</i>	<i>b</i>	<i>b</i>
	{ <i>X</i> , <i>A</i> }	{ <i>X</i> , <i>A</i> }	{ <i>Y</i> , <i>B</i> }	{ <i>X</i> , <i>A</i> }	{ <i>Y</i> , <i>B</i> }	{ <i>Y</i> , <i>B</i> }
	{ <i>X'</i> }	{ <i>S</i> }	{ <i>S</i> }	{ <i>S</i> }	{ <i>Y'</i> }	
	{ <i>X</i> }	{ <i>X</i> }	{ <i>Y</i> }	{ <i>Y</i> }		
	{ <i>X'</i> }					

# Der CYK-Algorithmus

## Beispiel (Fortsetzung)

- Betrachte die CNF-Grammatik mit den Regeln

$P: S \rightarrow AS', AY, BX, CS, c, S' \rightarrow BC, X \rightarrow AS, BX', a, X' \rightarrow XX,$   
 $Y \rightarrow BS, AY', b, Y' \rightarrow YY, A \rightarrow a, B \rightarrow b, C \rightarrow c.$

- Dagegen gehört das Wort  $y = aababb$  zu  $L(G)$ :

	<i>a</i>	<i>a</i>	<i>b</i>	<i>a</i>	<i>b</i>	<i>b</i>
	{X, A}	{X, A}	{Y, B}	{X, A}	{Y, B}	{Y, B}
	{X'}	{S}	{S}	{S}	{Y'}	
	{X}	{X}	{Y}	{Y}		
	{X'}	{S}				

# Der CYK-Algorithmus

## Beispiel (Fortsetzung)

- Betrachte die CNF-Grammatik mit den Regeln

$P: S \rightarrow AS', AY, BX, CS, c, \quad S' \rightarrow BC, \quad X \rightarrow AS, BX', a, \quad X' \rightarrow XX,$   
 $Y \rightarrow BS, AY', b, \quad Y' \rightarrow YY, \quad A \rightarrow a, \quad B \rightarrow b, \quad C \rightarrow c.$

- Dagegen gehört das Wort  $y = aababb$  zu  $L(G)$ :

	<i>a</i>	<i>a</i>	<i>b</i>	<i>a</i>	<i>b</i>	<i>b</i>
	{X, A}	{X, A}	{Y, B}	{X, A}	{Y, B}	{Y, B}
	{X'}	{S}	{S}	{S}	{Y'}	
	{X}	{X}	{Y}	{Y}		
	{X'}	{S}	{Y'}			

# Der CYK-Algorithmus

## Beispiel (Fortsetzung)

- Betrachte die CNF-Grammatik mit den Regeln

$P: S \rightarrow AS', AY, BX, CS, c, \quad S' \rightarrow BC, \quad X \rightarrow AS, BX', a, \quad X' \rightarrow XX,$   
 $Y \rightarrow BS, AY', b, \quad Y' \rightarrow YY, \quad A \rightarrow a, \quad B \rightarrow b, \quad C \rightarrow c.$

- Dagegen gehört das Wort  $y = aababb$  zu  $L(G)$ :

	<i>a</i>	<i>a</i>	<i>b</i>	<i>a</i>	<i>b</i>	<i>b</i>
	{X, <i>A</i> }	{X, A}	{Y, B}	{X, A}	{Y, B}	{Y, B}
	{X'}	{S}	{S}	{S}	{Y'}	
	{X}	{X}	{Y}	{Y}		
	{X'}	{ <i>S</i> }	{Y'}			
	{ <i>X</i> }					



# Der CYK-Algorithmus

## Beispiel (Fortsetzung)

- Betrachte die CNF-Grammatik mit den Regeln

$P: S \rightarrow AS', AY, BX, CS, c, \quad S' \rightarrow BC, \quad X \rightarrow AS, BX', a, \quad X' \rightarrow XX,$   
 $Y \rightarrow BS, AY', b, \quad Y' \rightarrow YY, \quad A \rightarrow a, \quad B \rightarrow b, \quad C \rightarrow c.$

- Dagegen gehört das Wort  $y = aababb$  zu  $L(G)$ :

	<i>a</i>	<i>a</i>	<i>b</i>	<i>a</i>	<i>b</i>	<i>b</i>
	{X, A}	{X, <i>A</i> }	{Y, B}	{X, A}	{Y, B}	{Y, B}
	{X'}	{S}	{S}	{S}	{Y'}	
	{X}	{X}	{Y}	{Y}		
	{X'}	{S}	{ <i>Y'</i> }			
	{X}	{ <i>Y</i> }				

# Der CYK-Algorithmus

## Beispiel (Fortsetzung)

- Betrachte die CNF-Grammatik mit den Regeln

$P: S \rightarrow AS', AY, BX, CS, c, S' \rightarrow BC, X \rightarrow AS, BX', a, X' \rightarrow XX, Y \rightarrow BS, AY', b, Y' \rightarrow YY, A \rightarrow a, B \rightarrow b, C \rightarrow c.$

- Dagegen gehört das Wort  $y = aababb$  zu  $L(G)$ :

	<i>a</i>	<i>a</i>	<i>b</i>	<i>a</i>	<i>b</i>	<i>b</i>
	{X, A}	{X, A}	{Y, B}	{X, A}	{Y, B}	{Y, B}
	{X'}	{S}	{S}	{S}	{Y'}	
	{X}	{X}	{Y}	{Y}		
	{X'}	{S}	{Y'}			
	{X}	{Y}				
	{S}					

# Ein Maschinenmodell für die kontextfreien Sprachen

## Frage

Wie lässt sich das Maschinenmodell des DFA erweitern, um die Sprache

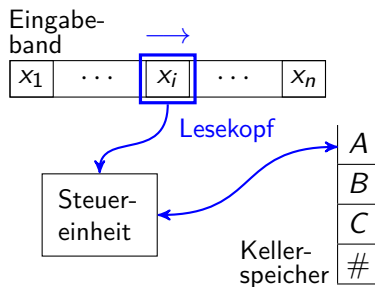
$$L = \{a^n b^n \mid n \geq 0\}$$

und alle anderen kontextfreien Sprachen erkennen zu können?

## Antwort

- Dass ein DFA die Sprache  $L$  nicht erkennen kann, liegt an seinem beschränkten Speichervermögen, das zwar von  $L$  aber nicht von der Eingabe abhängen darf.
- Um  $L$  erkennen zu können, genügt bereits ein so genannter **Kellerspeicher** (auch **Stapel**, engl. *stack* oder *pushdown memory*).
- Dieser erlaubt nur den Zugriff auf die höchste belegte Speicheradresse.

# Der Kellerautomat



- verfügt zusätzlich über einen Kellerspeicher,
- kann auch  $\varepsilon$ -Übergänge machen,
- hat Lesezugriff auf das aktuelle Eingabezeichen und auf das oberste Kellersymbol,
- kann das oberste Kellersymbol löschen (durch eine **pop-Operation**) und
- danach beliebig viele Symbole einkellern (mittels **push-Operationen**).

# Formale Definition des Kellerautomaten

## Notation

Für eine Menge  $M$  bezeichne  $\mathcal{P}_e(M)$  die Menge aller endlichen Teilmengen von  $M$ , d.h.

$$\mathcal{P}_e(M) = \{A \subseteq M \mid A \text{ ist endlich}\}.$$

## Definition

Ein **Kellerautomat** (kurz: **PDA**, engl. *pushdown automaton*) wird durch ein 6-Tupel  $M = (Z, \Sigma, \Gamma, \delta, q_0, \#)$  beschrieben, wobei

- $Z \neq \emptyset$  eine endliche Menge von **Zuständen**,
- $\Sigma$  das **Eingabealphabet**,
- $\Gamma$  das **Kelleralphabet**,
- $\delta : Z \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \rightarrow \mathcal{P}_e(Z \times \Gamma^*)$  die **Überföhrungsfunktion**,
- $q_0 \in Z$  der **Startzustand** und
- $\# \in \Gamma$  das **Kelleranfangszeichen** ist.

# Der Kellerautomat

## Arbeitsweise eines PDA

- Wenn  $p$  der momentane Zustand,  $A$  das oberste Kellerzeichen und  $u \in \Sigma$  das nächste Eingabezeichen (bzw.  $u = \varepsilon$ ) ist, so kann  $M$  im Fall  $(q, B_1 \cdots B_k) \in \delta(p, u, A)$ 
  - in den Zustand  $q$  wechseln,
  - den Lesekopf auf dem Eingabeband um  $|u| \in \{0, 1\}$  Positionen vorrücken und
  - das Zeichen  $A$  aus- sowie die Zeichenfolge  $B_1 \cdots B_k$  einkellern (danach ist  $B_1$  das oberste Kellerzeichen).
- Hierfür sagen wir auch,  $M$  führt die **Anweisung**
$$puA \rightarrow qB_1 \cdots B_k$$
aus.
- Im Fall  $u = \varepsilon$  spricht man auch von einem  **$\varepsilon$ -Übergang**.

# Formale Definition der Konfiguration eines PDA

- Eine **Konfiguration** wird durch ein Tripel

$$K = (p, x_i \cdots x_n, A_1 \cdots A_l) \in Z \times \Sigma^* \times \Gamma^*$$

beschrieben und besagt, dass

- $p$  der momentane Zustand,
  - $x_i \cdots x_n$  der ungelesene Rest der Eingabe und
  - $A_1 \cdots A_l$  der aktuelle Kellerinhalt ist ( $A_1$  ist oberstes Symbol).
- In der Konfiguration  $K = (p, x_i \cdots x_n, A_1 \cdots A_l)$  kann  $M$  eine bel. Anweisung  $puA_1 \rightarrow qB_1 \cdots B_k$  mit  $u \in \{\varepsilon, x_i\}$  ausführen.

Diese überführt  $M$  in die **Folgekonfiguration**

$$K' = (q, x_j \cdots x_n, B_1 \cdots B_k A_2 \cdots A_l) \text{ mit } j = i + |u|.$$

Hierfür schreiben wir auch kurz  $K \vdash K'$ .

- Eine **Rechnung** von  $M$  bei Eingabe  $x$  ist eine Folge von Konfigurationen  $K_0, K_1, K_2 \dots$  mit  $K_0 = (q_0, x, \#)$  und  $K_0 \vdash K_1 \vdash K_2 \dots$ .  
 $K_0$  heißt **Startkonfiguration** von  $M$  bei Eingabe  $x$ .

# Definition der von einem PDA erkannten Sprache

## Notation

Die reflexive, transitive Hülle von  $\vdash$  bezeichnen wir wie üblich mit  $\vdash^*$ .

## Definition

Die von  $M = (Z, \Sigma, \Gamma, \delta, q_0, \#)$  **akzeptierte** oder **erkannte Sprache** ist

$$L(M) = \{x \in \Sigma^* \mid \exists q \in Z : (q_0, x, \#) \vdash^* (q, \varepsilon, \varepsilon)\}.$$

## Bemerkung

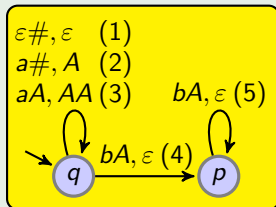
- Ein PDA  $M$  akzeptiert also genau dann eine Eingabe  $x$ , wenn es eine Rechnung gibt, bei der  $M$ 
  - das gesamte Eingabewort bis zum Ende liest und
  - den Keller leert.
- Man beachte, dass bei leerem Keller kein weiterer Übergang mehr möglich ist.



# Ein Kellerautomat

## Beispiel

- Sei  $M = (Z, \Sigma, \Gamma, \delta, q, \#)$  mit  $Z = \{q, p\}$ ,  
 $\Sigma = \{a, b\}$ ,  $\Gamma = \{A, \#\}$  und  
 $\delta : q\varepsilon\# \rightarrow q$  (1)  $qa\# \rightarrow qA$  (2)  $qaA \rightarrow qAA$  (3)  
 $qbA \rightarrow p$  (4)  $pbA \rightarrow p$  (5)



- Dann akzeptiert  $M$  die Eingabe  $x = aabb$ :

$$(q, aabb, \#) \underset{(2)}{\vdash} (q, abb, A) \underset{(3)}{\vdash} (q, bb, AA) \underset{(4)}{\vdash} (p, b, A) \underset{(5)}{\vdash} (p, \varepsilon, \varepsilon).$$

- Allgemeiner akzeptiert  $M$  das Wort  $x = a^n b^n$  mit folgender Rechnung:

$$n = 0: (q, \varepsilon, \#) \underset{(1)}{\vdash} (q, \varepsilon, \varepsilon).$$

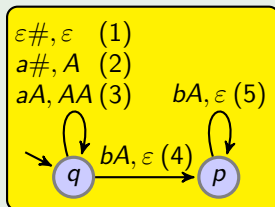
$$n \geq 1: (q, a^n b^n, \#) \underset{(2)}{\vdash} (q, a^{n-1} b^n, A) \underset{(3)}{\vdash}^{n-1} (q, b^n, A^n) \\ \underset{(4)}{\vdash} (p, b^{n-1}, A^{n-1}) \underset{(5)}{\vdash}^{n-1} (p, \varepsilon, \varepsilon).$$

- Dies zeigt, dass  $M$  alle Wörter der Form  $a^n b^n$ ,  $n \geq 0$ , akzeptiert.

# Ein Kellerautomat

## Beispiel

- Sei  $M = (Z, \Sigma, \Gamma, \delta, q, \#)$  mit  $Z = \{q, p\}$ ,  
 $\Sigma = \{a, b\}$ ,  $\Gamma = \{A, \#\}$  und  
 $\delta : q\varepsilon\# \rightarrow q$  (1)  $qa\# \rightarrow qA$  (2)  $qaA \rightarrow qAA$  (3)  
 $qbA \rightarrow p$  (4)  $pbA \rightarrow p$  (5)

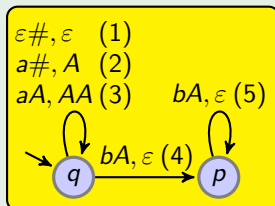


- Als nächstes zeigen wir, dass jede von  $M$  akzeptierte Eingabe  $x = x_1 \dots x_n \in L(M)$  die Form  $x = a^m b^m$  haben muss.
- Ausgehend von der Startkonfiguration  $(q, x, \#)$  sind nur die Anweisungen (1) oder (2) ausführbar.
- Führt  $M$  zuerst Anweisung (1) aus, so wird der Keller geleert.
- Daher kann  $M$  in diesem Fall nur das leere Wort  $x = \varepsilon = a^0 b^0$  akzeptieren.
- Falls  $M$  mit Anweisung (2) beginnt, muss  $M$  später mittels Anweisung (4) in den Zustand  $p$  gelangen, da sonst der Keller nicht geleert wird.

# Ein Kellerautomat

## Beispiel

- Sei  $M = (Z, \Sigma, \Gamma, \delta, q, \#)$  mit  $Z = \{q, p\}$ ,  
 $\Sigma = \{a, b\}$ ,  $\Gamma = \{A, \#\}$  und  
 $\delta : q\varepsilon\# \rightarrow q$  (1)  $qa\# \rightarrow qA$  (2)  $qaA \rightarrow qAA$  (3)  
 $qbA \rightarrow p$  (4)  $pbA \rightarrow p$  (5)



- Falls  $M$  mit Anweisung (2) beginnt, muss  $M$  später mittels Anweisung (4) in den Zustand  $p$  gelangen, da sonst der Keller nicht geleert wird.
- Dies geschieht, sobald  $M$  nach Lesen von  $m \geq 1$   $a$ 's das erste  $b$  liest:

$$(q, x_1 \cdots x_n, \#) \underset{(2)}{\vdash} (q, x_2 \cdots x_n, A) \underset{(3)}{\vdash}^{m-1} (q, x_{m+1} \cdots x_n, A^m) \\ \underset{(4)}{\vdash} (p, x_{m+2} \cdots x_n, A^{m-1})$$

mit  $x_1 = x_2 = \cdots = x_m = a$  und  $x_{m+1} = b$ .

- Um den Keller leeren zu können, muss  $M$  nun noch genau  $m - 1$   $b$ 's lesen, weshalb  $x$  auch in diesem Fall die Form  $a^m b^m$  haben muss. ◀

# Ein Maschinenmodell für die Klasse CFL

## Ziel

Als nächstes wollen wir zeigen, dass PDAs genau die kontextfreien Sprachen erkennen.

## Satz

$CFL = \{L(M) \mid M \text{ ist ein PDA}\}.$

## Beweis von $\text{CFL} \subseteq \{L(M) \mid M \text{ ist ein PDA}\}$

### Idee:

- Konstruiere zu einer kontextfreien Grammatik  $G = (V, \Sigma, P, S)$  einen PDA  $M = (\{q\}, \Sigma, \Gamma, \delta, q, S)$  mit  $\Gamma = V \cup \Sigma$ , so dass folgende Äquivalenz gilt:

$$S \Rightarrow^* x_1 \cdots x_n \text{ gdw. } (q, x_1 \cdots x_n, S) \vdash^* (q, \varepsilon, \varepsilon).$$

- Hierzu fügen wir folgende Anweisungen zu  $\delta$  hinzu:

für jede Regel  $A \rightarrow_G \alpha$ :  $q\varepsilon A \rightarrow q\alpha$ ,

für jedes Zeichen  $a \in \Sigma$ :  $qaa \rightarrow q\varepsilon$ .

- $M$  versucht also, eine Linksableitung für die Eingabe  $x$  zu finden.
- Da  $M$  hierbei den Syntaxbaum von oben nach unten aufbaut, wird  $M$  als *Top-Down Parser* bezeichnet.
- Dann gilt  $S \Rightarrow_L^* x_1 \cdots x_n$  gdw.  $(q, x_1 \cdots x_n, S) \vdash^{l+n} (q, \varepsilon, \varepsilon)$ .
- Daher folgt

$$x \in L(G) \Leftrightarrow S \Rightarrow_L^* x \Leftrightarrow (q, x, S) \vdash^* (q, \varepsilon, \varepsilon) \Leftrightarrow x \in L(M). \quad \square$$

# Beweis von $CFL \subseteq \{L(M) \mid M \text{ ist ein PDA}\}$

## Beispiel

- Betrachte die Grammatik  $G = (\{S\}, \{a, b\}, P, S)$  mit den Regeln

$$P: S \rightarrow aSbS \quad (1), \quad S \rightarrow a \quad (2).$$

- Der zugehörige PDA besitzt dann die Anweisungen

$$\delta : qaa \rightarrow q\varepsilon \quad (0) \quad qbb \rightarrow q\varepsilon \quad (0')$$

$$q\varepsilon S \rightarrow qaSbS \quad (1') \quad q\varepsilon S \rightarrow qa \quad (2')$$

- Der Linksableitung  $\underline{S} \xRightarrow{(1)} a\underline{S}bS \xRightarrow{(2)} aab\underline{S} \xRightarrow{(2)} aaba$  in  $G$  entspricht dann die Rechnung

$$(q, aaba, S) \underset{(1')}{\vdash} (q, aaba, aSbS) \underset{(0)}{\vdash} (q, aba, SbS) \underset{(2')}{\vdash} (q, aba, abS)$$

$$\underset{(0)}{\vdash} (q, ba, bS) \underset{(0')}{\vdash} (q, a, S) \underset{(2')}{\vdash} (q, a, a) \underset{(0)}{\vdash} (q, \varepsilon, \varepsilon)$$

von  $M$  und umgekehrt.

## Beweis von $\{L(M) \mid M \text{ ist ein PDA}\} \subseteq \text{CFL}$

### Idee:

- Konstruiere zu einem PDA  $M = (Z, \Sigma, \Gamma, \delta, q_0, \#)$  eine kontextfreie Grammatik  $G = (V, \Sigma, P, S)$  mit Variablen  $X_{pAp'}$ ,  $A \in \Gamma$ ,  $p, p' \in Z$ , so dass folgende Äquivalenz gilt:

$$(p, x, A) \vdash^* (p', \varepsilon, \varepsilon) \text{ gdw. } X_{pAp'} \Rightarrow^* x.$$

- Ein Wort  $x$  soll also genau dann in  $G$  aus  $X_{pAp'}$  ableitbar sein, wenn  $M$  ausgehend vom Zustand  $p$  bei Lesen von  $x$  in den Zustand  $p'$  gelangen kann und dabei das Zeichen  $A$  aus dem Keller entfernt.
- Hierzu fügen wir für jede Anweisung  $puA \rightarrow p_0A_1 \cdots A_k$ ,  $k \geq 0$ , die folgenden  $\|Z\|^k$  Regeln zu  $P$  hinzu:

$$\text{Für jede Zustandsfolge } p_1, \dots, p_k: X_{pAp_k} \rightarrow uX_{p_0A_1p_1} \cdots X_{p_{k-1}A_kp_k}.$$

- Um damit alle Wörter  $x \in L(M)$  aus  $S$  ableiten zu können, benötigen wir jetzt nur noch die Regeln

$$S \rightarrow X_{q_0\#p'}, p' \in Z.$$

# Beweis von $\{L(M) \mid M \text{ ist ein PDA}\} \subseteq \text{CFL}$

## Beispiel

- Betrachte den PDA  $M = (\{p, q\}, \{a, b\}, \{A, \#\}, \delta, p, \#)$  mit den Anweisungen

$$\begin{array}{lll} \delta : p\varepsilon\# \rightarrow q\varepsilon & (1) & pa\# \rightarrow pA & (2) & paA \rightarrow pAA & (3) \\ & & pbA \rightarrow q\varepsilon & (4) & qbA \rightarrow q\varepsilon & (5) \end{array}$$

- Dann erhalten wir die Grammatik  $G = (V, \Sigma, P, S)$  mit der Variablenmenge

$$V = \{S, X_{p\#p}, X_{p\#q}, X_{q\#p}, X_{q\#q}, X_{pAp}, X_{pAq}, X_{qAp}, X_{qAq}\}.$$

- Die Regelmenge  $P$  enthält neben den beiden Startregeln

$$S \rightarrow X_{p\#p}, X_{p\#q} \quad (0, 0')$$

die folgenden Produktionen:



# Beweis von $\{L(M) \mid M \text{ ist ein PDA}\} \subseteq \text{CFL}$

## Beispiel (Fortsetzung)

- $P$  enthält neben den beiden Startregeln  $S \rightarrow X_{p\#p}, X_{p\#q}$  ( $0, 0'$ ) die folgenden Produktionen:

Anweisung	$k$	$p_1, \dots, p_k$	zugehörige Regeln
$p\epsilon\# \rightarrow q\epsilon$	(1)	0	$X_{p\#q} \rightarrow \epsilon$ (1')
$pa\# \rightarrow pA$	(2)	1	$X_{p\#p} \rightarrow aX_{pAp}$ (2') $X_{p\#q} \rightarrow aX_{pAq}$ (2'')
$paA \rightarrow pAA$	(3)	2	$X_{pAp} \rightarrow aX_{pAp}X_{pAp}$ (3') $X_{pAq} \rightarrow aX_{pAp}X_{pAq}$ (3'') $X_{pAp} \rightarrow aX_{pAq}X_{qAp}$ (3''') $X_{pAq} \rightarrow aX_{pAq}X_{qAq}$ (3'''' )
$pbA \rightarrow q\epsilon$	(4)	0	$X_{pAq} \rightarrow b$ (4')
$qbA \rightarrow q\epsilon$	(5)	0	$X_{qAq} \rightarrow b$ (5')

# Beweis von $\{L(M) \mid M \text{ ist ein PDA}\} \subseteq \text{CFL}$

## Beispiel (Schluss)

- Die Anweisungen

$$\begin{array}{lll} \delta : p\varepsilon\# \rightarrow q\varepsilon & (1) & pa\# \rightarrow pA \quad (2) \quad paA \rightarrow pAA \quad (3) \\ & & pbA \rightarrow q\varepsilon \quad (4) \quad qbA \rightarrow q\varepsilon \quad (5) \end{array}$$

von  $M$  führen also auf die folgenden Regeln von  $G$ :

$$\begin{array}{lll} S \rightarrow X_{p\#p}, X_{p\#q} & (0, 0') & X_{p\#q} \rightarrow \varepsilon \quad (1') \\ X_{p\#p} \rightarrow aX_{pAp} & (2') & X_{p\#q} \rightarrow aX_{pAq} \quad (2'') \\ X_{pAp} \rightarrow aX_{pAp}X_{pAp} & (3') & X_{pAq} \rightarrow aX_{pAp}X_{pAq} \quad (3''') \\ X_{pAp} \rightarrow aX_{pAq}X_{qAp} & (3''') & X_{pAq} \rightarrow aX_{pAq}X_{qAq} \quad (3''''') \\ X_{pAq} \rightarrow b & (4') & X_{qAq} \rightarrow b \quad (5') \end{array}$$

- Der akzeptierenden Rechnung

$$(p, aabb, \#) \underset{(2)}{\vdash} (p, abb, A) \underset{(3)}{\vdash} (p, bb, AA) \underset{(4)}{\vdash} (q, b, A) \underset{(5)}{\vdash} (q, \varepsilon, \varepsilon)$$

von  $M$  entspricht dann in  $G$  die Linksableitung

$$\underline{S} \underset{(0')}{\Rightarrow} \underline{X_{p\#q}} \underset{(2'')}{\Rightarrow} \underline{aX_{pAq}} \underset{(3''''')}{\Rightarrow} \underline{aaX_{pAq}X_{qAq}} \underset{(4')}{\Rightarrow} \underline{aabX_{qAq}} \underset{(5')}{\Rightarrow} \underline{aabb}.$$

## Beweis von $\{L(M) \mid M \text{ ist ein PDA}\} \subseteq \text{CFL}$

- Für einen PDA  $M = (Z, \Sigma, \Gamma, \delta, q_0, \#)$  sei  $G$  die Grammatik  $(V, \Sigma, P, S)$  mit  $V = \{S\} \cup \{X_{pAq} \mid p, q \in Z, A \in \Gamma\}$ , wobei  $P$  neben den Regeln  $S \rightarrow X_{q_0\#p'}$ ,  $p' \in Z$ , für jede Anweisung

$$p u A \rightarrow p_0 A_1 \cdots A_k, \quad k \geq 0$$

von  $M$  und jede Zustandsfolge  $p_1, \dots, p_k$  die folgende Regel enthält:

$$X_{pA p_k} \rightarrow u X_{p_0 A_1 p_1} \cdots X_{p_{k-1} A_k p_k}.$$

- Dann lässt sich mit Hilfe der Äquivalenz

$$(p, x, A) \vdash^* (p', \varepsilon, \varepsilon) \text{ gdw. } X_{pAp'} \Rightarrow^* x,$$

deren Beweis wir später nachholen, leicht die Korrektheit von  $G$  zeigen:

$$x \in L(M) \Leftrightarrow (q_0, x, \#) \vdash^* (p', \varepsilon, \varepsilon) \text{ für ein } p' \in Z$$

$$\Leftrightarrow S \Rightarrow X_{q_0\#p'} \Rightarrow^* x \text{ für ein } p' \in Z$$

$$\Leftrightarrow x \in L(G).$$

## Beweis von $(p, x, A) \vdash^* (p', \varepsilon, \varepsilon)$ gdw. $X_{pAp'} \Rightarrow^* x$

- Es bleibt zu zeigen, dass für alle  $p, p' \in Z$ ,  $A \in \Gamma$  und  $x \in \Sigma^*$  folgende Äquivalenz gilt:

$$(p, x, A) \vdash^* (p', \varepsilon, \varepsilon) \text{ gdw. } X_{pAp'} \Rightarrow^* x. \quad (*)$$

- Hierzu zeigen wir durch Induktion über  $m$  folgende stärkere Behauptung:

$$(p, x, A) \vdash^m (p', \varepsilon, \varepsilon) \text{ gdw. } X_{pAp'} \Rightarrow^m x. \quad (**)$$

### Beweis von **(\*\*)** durch Induktion über $m$ :

$m = 0$ : Da weder  $(p, x, A) \vdash^0 (p', \varepsilon, \varepsilon)$  noch  $X_{pAp'} \Rightarrow^0 x$  gelten, ist die Äquivalenz **(\*\*)** für  $m = 0$  erfüllt.

## Beweis von $(p, x, A) \vdash^m (p', \varepsilon, \varepsilon)$ gdw. $X_{pAp'} \Rightarrow^m x$

$m \rightsquigarrow m + 1$ : Wir zeigen zuerst die Implikation von links nach rechts.

- Sei eine Rechnung der Länge  $m + 1$  gegeben:

$$(p, x, A) \vdash (p_0, x', A_1 \cdots A_k) \vdash^m (p', \varepsilon, \varepsilon).$$

- Sei  $puA \rightarrow p_0A_1 \cdots A_k$ ,  $k \geq 0$ , die im ersten Rechenschritt ausgeführte Anweisung (d.h.  $x = ux'$ ).
- Für  $i = 1, \dots, k$  sei  $p_i$  der Zustand, in den  $M$  mit Kellerinhalt  $A_{i+1} \cdots A_k$  gelangt (d.h.  $p_k = p'$ ).
- Dann enthält  $P$  die Regel  $X_{pAp_k} \rightarrow uX_{p_0A_1p_1} \cdots X_{p_{k-1}A_kp_k}$ .
- Zudem sei  $u_i$  für  $i = 1, \dots, k$  das Teilwort von  $x'$ , das  $M$  zwischen den Besuchen von  $p_{i-1}$  und  $p_i$  liest.
- Dann ex. Zahlen  $m_i \geq 1$  mit  $m_1 + \cdots + m_k = m$  und

$$(p_i, u_{i+1}, A_{i+1}) \vdash^{m_{i+1}} (p_{i+1}, \varepsilon, \varepsilon) \text{ f\"ur } i = 0, \dots, k - 1.$$

- Nach IV gibt es daher Ableitungen

$$X_{p_iA_{i+1}p_{i+1}} \Rightarrow^{m_{i+1}} u_{i+1}, \quad i = 0, \dots, k - 1.$$

# Beweis von $(p, x, A) \vdash^m (p', \varepsilon, \varepsilon)$ gdw. $X_{pAp'} \Rightarrow^m x$

## Induktionsschritt für die Implikation von links nach rechts

- Nach IV gibt es daher Ableitungen

$$X_{p_i A_{i+1} p_{i+1}} \Rightarrow^{m_{i+1}} u_{i+1}, \quad i = 0, \dots, k-1,$$

die wir zu der gesuchten Ableitung zusammensetzen können:

$$\begin{aligned} X_{pAp_k} &\Rightarrow && uX_{p_0 A_1 p_1} \cdots X_{p_{k-2} A_{k-1} p_{k-1}} X_{p_{k-1} A_k p_k} \\ &\Rightarrow^{m_1} && uu_1 X_{p_1 A_2 p_2} \cdots X_{p_{k-2} A_{k-1} p_{k-1}} X_{p_{k-1} A_k p_k} \\ &&& \vdots \\ &\Rightarrow^{m_{k-1}} && uu_1 \cdots u_{k-1} X_{p_{k-1} A_k p_k} \\ &\Rightarrow^{m_k} && uu_1 \cdots u_k = x. \end{aligned}$$

## Beweis von $(p, x, A) \vdash^m (p', \varepsilon, \varepsilon)$ gdw. $X_{pAp'} \Rightarrow^m x$

### Induktionsschritt für die Implikation von rechts nach links

- Gelte nun umgekehrt  $X_{pAp'} \Rightarrow^{m+1} x$  und sei  $\alpha$  die im ersten Schritt abgeleitete Satzform, d.h.  $X_{pAp'} \Rightarrow \alpha \Rightarrow^m x$ .
- Wegen  $X_{pAp'} \rightarrow_G \alpha$  gibt es eine Anweisung  $puA \rightarrow p_0A_1 \cdots A_k$ ,  $k \geq 0$ , und Zustände  $p_1, \dots, p_k \in Z$  mit

$$\alpha = u X_{p_0A_1p_1} \cdots X_{p_{k-1}A_kp_k}, \text{ wobei } p_k = p' \text{ ist.}$$

- Wegen  $\alpha \Rightarrow^m x$  ex. eine Zerlegung  $x = uu_1 \cdots u_k$  und Zahlen  $m_i \geq 1$  mit  $m_1 + \cdots + m_k = m$  und

$$X_{p_iA_{i+1}p_{i+1}} \Rightarrow^{m_i+1} u_{i+1} \quad (i = 0, \dots, k-1).$$

- Nach IV gibt es somit Rechnungen

$$(p_i, u_{i+1}, A_{i+1}) \vdash^{m_i+1} (p_{i+1}, \varepsilon, \varepsilon), \quad i = 0, \dots, k-1.$$

Beweis von  $(p, x, A) \vdash^m (p', \varepsilon, \varepsilon)$  gdw.  $X_{pAp'} \Rightarrow^m x$

Induktionsschritt für die Implikation von rechts nach links

- Nach IV gibt es somit Rechnungen

$$(p_i, u_{i+1}, A_{i+1}) \vdash^{m_{i+1}} (p_{i+1}, \varepsilon, \varepsilon), \quad i = 0, \dots, k-1,$$

aus denen sich die gesuchte Rechnung der Länge  $m+1$  zusammensetzen lässt:

$$\begin{aligned} (p, uu_1 \cdots u_k, A) \vdash & (p_0, u_1 \cdots u_k, A_1 \cdots A_k) \\ & \vdash^{m_1} (p_1, u_2 \cdots u_k, A_2 \cdots A_k) \\ & \vdots \\ & \vdash^{m_{k-1}} (p_{k-1}, u_k, A_k) \\ & \vdash^{m_k} (p_k, \varepsilon, \varepsilon). \end{aligned}$$





# Deterministische Kellerautomaten

Von besonderem Interesse sind kontextfreie Sprachen, die von einem deterministischen Kellerautomaten erkannt werden.

## Definition

- Ein Kellerautomat heißt **deterministisch**, falls  $\vdash$  eine rechtseindeutige Relation ist:

$$K \vdash K_1 \wedge K \vdash K_2 \Rightarrow K_1 = K_2.$$

- Äquivalent hierzu ist, dass die Überföhrungsfunktion  $\delta$  für alle  $(q, a, A) \in Z \times \Sigma \times \Gamma$  folgende Bedingung erfüllt (siehe Übungen):

$$\|\delta(q, a, A)\| + \|\delta(q, \varepsilon, A)\| \leq 1.$$

# Deterministische Kellerautomaten

## Beispiel

- Betrachte den PDA  $M = (\{q_0, q_1, q_2\}, \{a, b, c\}, \{A, B, \#\}, \delta, q_0, \#)$  mit  
 $\delta$ :  $q_0 a \# \rightarrow q_0 A \#$      $q_0 b \# \rightarrow q_0 B \#$      $q_0 a A \rightarrow q_0 AA$      $q_0 b A \rightarrow q_0 BA$   
 $q_0 a B \rightarrow q_0 AB$      $q_0 b B \rightarrow q_0 BB$      $q_0 c A \rightarrow q_1 A$      $q_0 c B \rightarrow q_1 B$   
 $q_1 a A \rightarrow q_1$      $q_1 b B \rightarrow q_1$      $q_1 \varepsilon \# \rightarrow q_2$

Darstellung von  $\delta$  in Tabellenform

$\delta$	$q_0, \#$	$q_0, A$	$q_0, B$	$q_1, \#$	$q_1, A$	$q_1, B$	$q_2, \#$	$q_2, A$	$q_2, B$
$\varepsilon$	—	—	—	$q_2$	—	—	—	—	—
$a$	$q_0 A \#$	$q_0 AA$	$q_0 AB$	—	$q_1$	—	—	—	—
$b$	$q_0 B \#$	$q_0 BA$	$q_0 BB$	—	—	$q_1$	—	—	—
$c$	—	$q_1 A$	$q_1 B$	—	—	—	—	—	—

- Man beachte, dass jedes Tabellenfeld höchstens eine Anweisung enthält und jede Spalte mit einem  $\varepsilon$ -Übergang keine weiteren Anweisungen enthält.
- Daher ist die Bedingung  $\|\delta(q, a, A)\| + \|\delta(q, \varepsilon, A)\| \leq 1$  für alle  $q \in Z$ ,  $a \in \Sigma$  und  $A \in \Gamma$  erfüllt.

# Deterministische Kellerautomaten

## Frage

- Können deterministische Kellerautomaten zumindest alle regulären Sprachen durch Leeren des Kellers akzeptieren?
- Kann z.B. die Sprache  $L = \{a, aa\}$  von einem deterministischen Kellerautomaten  $M$  durch Leeren des Kellers akzeptiert werden?

## Antwort: Nein

- Um  $x = a$  zu akzeptieren, muss  $M$  den Keller nach Lesen von  $a$  leeren und kann somit keine anderen Wörter mit dem Präfix  $a$  akzeptieren.
- Deterministische Kellerautomaten können also durch Leeren des Kellers nur **präfixfreie** Sprachen  $L$  akzeptieren (d.h. kein Wort  $x \in L$  ist Präfix eines anderen Wortes in  $L$ ).

## Lösung des Problems

Wir vereinbaren, dass deterministische Kellerautomaten ihre Eingabe durch Erreichen eines Endzustands akzeptieren dürfen.

# Deterministische Kellerautomaten

## Definition

- Ein **Kellerautomat mit Endzuständen** wird durch ein 7-Tupel  $M = (Z, \Sigma, \Gamma, \delta, q_0, \#, E)$  beschrieben.
- Dabei sind die Komponenten  $Z, \Sigma, \Gamma, \delta, q_0, \#$  wie bei einem PDA.
- Zusätzlich ist  $E \subseteq Z$  eine Menge von **Endzuständen**.
- Die von  $M$  **akzeptierte** oder **erkannte Sprache** ist

$$L(M) = \{x \in \Sigma^* \mid \exists p \in E \exists \alpha \in \Gamma^* : (q_0, x, \#) \vdash^* (p, \varepsilon, \alpha)\}.$$

- $M$  ist ein **det. Kellerautomat mit Endzuständen** (kurz: **DPDA**), falls  $M$  für alle  $(q, a, A) \in Z \times \Sigma \times \Gamma$  zusätzlich folgende Bedingung erfüllt:

$$\|\delta(q, a, A)\| + \|\delta(q, \varepsilon, A)\| \leq 1.$$

- Weiter sei

$$\text{DCFL} = \{L(M) \mid M \text{ ist ein DPDA}\}$$

(*deterministic context free languages*).

## Charakterisierung von DCFL mittels Grammatiken

- Die Klasse DCFL lässt sich auch mit Hilfe von speziellen kontextfreien Grammatiken charakterisieren, den so genannten  $LR(k)$ -Grammatiken.
- Der erste Buchstabe  $L$  steht für die Leserichtung bei der Syntaxanalyse, d.h. das Eingabewort  $x$  wird von links (nach rechts) gelesen.
- Der zweite Buchstabe  $R$  bedeutet, dass bei der Syntaxanalyse eine Rechtsableitung entsteht.
- Schließlich gibt der Parameter  $k$  an, wieviele Zeichen man in der Eingabe vorauslesen muss, damit die nächste anzuwendende Regel eindeutig feststeht ( $k$  wird auch als *Lookahead* bezeichnet).
- Durch  $LR(0)$ -Grammatiken lassen sich nur die präfixfreien Sprachen in DCFL erzeugen.
- Dagegen erzeugen die  $LR(k)$ -Grammatiken für jedes  $k \geq 1$  genau die Sprachen in DCFL.
- Daneben gibt es noch  $LL(k)$ -Grammatiken, die für wachsendes  $k$  immer mehr deterministisch kontextfreie Sprachen erzeugen.

# Abschlusseigenschaften von DCFL

## Frage

Ist DCFL unter Komplementbildung abgeschlossen?

## Antwort

Ja. Allerdings ergeben sich beim Versuch, einfach die End- und Nicht-endzustände eines DPDA  $M$  zu vertauschen, um einen DPDA  $\overline{M}$  für  $\overline{L(M)}$  zu erhalten, folgende Schwierigkeiten:

- 1 Falls  $M$  eine Eingabe  $x$  nicht zu Ende liest, wird  $x$  weder von  $M$  noch von  $\overline{M}$  akzeptiert.
- 2 Falls  $M$  nach dem Lesen von  $x$  noch  $\varepsilon$ -Übergänge ausführt und dabei End- und Nichtendzustände besucht, wird  $x$  von  $M$  und von  $\overline{M}$  akzeptiert.

## DPDAs, die ihre Eingabe zu Ende lesen

Der nächste Satz zeigt, wie sich Problem 1 beheben lässt.

### Satz

Jede Sprache  $L \in \text{DCFL}$  wird von einem DPDA  $M'$  erkannt, der alle Eingaben zu Ende liest.

### Beweis.

Sei  $M = (Z, \Sigma, \Gamma, \delta, q_0, \#, E)$  ein DPDA mit  $L(M) = L$ .

Falls  $M$  eine Eingabe  $x = x_1 \cdots x_n$  nicht zu Ende liest, muss einer der folgenden drei Gründe vorliegen:

- 1  $M$  gerät in eine Konfiguration  $(q, x_i \cdots x_n, \varepsilon)$ ,  $i \leq n$ , mit leerem Keller.
- 2  $M$  gerät in eine Konfiguration  $(q, x_i \cdots x_n, A\gamma)$ ,  $i \leq n$ , in der wegen  $\delta(q, x_i, A) = \delta(q, \varepsilon, A) = \emptyset$  keine Anweisung ausführbar ist.
- 3  $M$  gerät in eine Konfiguration  $(q, x_i \cdots x_n, A\gamma)$ ,  $i \leq n$ , so dass  $M$  ausgehend von  $(q, \varepsilon, A)$  eine unendliche Folge von  $\varepsilon$ -Anweisungen ausführt.

## DPDAs, die ihre Eingabe zu Ende lesen

### Beweis (Fortsetzung)

- Die erste Ursache schließen wir aus, indem wir ein neues Zeichen  $\square$  auf dem Kellerboden platzieren:
  - (a)  $s\varepsilon\# \rightarrow q_0\#\square$  (dabei sei  $s$  ein neuer Startzustand).
- Die zweite Ursache schließen wir durch Hinzunahme eines Fehlerzustands  $f$  sowie folgender Anweisungen aus:
  - (b)  $qaA \rightarrow fA$ , für alle  $(q, a, A) \in Z \times \Sigma \times \Gamma'$  mit  $A = \square$  oder  $\delta(q, a, A) = \delta(q, \varepsilon, A) = \emptyset$  (hierbei ist  $\Gamma' = \Gamma \cup \{\square\}$ ),
  - (c)  $faA \rightarrow fA$ , für alle  $a \in \Sigma$  und  $A \in \Gamma'$ .



# DPDAs, die ihre Eingabe zu Ende lesen

## Beweis (Fortsetzung)

- Als nächstes verhindern wir die Ausführung einer unendlichen Folge von  $\varepsilon$ -Übergängen.

Dabei unterscheiden wir die beiden Fälle, ob  $M$  hierbei auch Endzustände besucht oder nicht.

(d)  $q\varepsilon A \rightarrow fA$ , für alle  $q \in Z$  und  $A \in \Gamma$ , so dass  $M$  ausgehend von der Konfiguration  $(q, \varepsilon, A)$  unendlich viele  $\varepsilon$ -Übergänge ausführt ohne dabei einen Endzustand zu besuchen.

Falls ja, sehen wir einen Umweg über den neuen Endzustand  $e$  vor.

(e)  $q\varepsilon A \rightarrow eA$  für alle  $q \in Z$  und  $A \in \Gamma$ , so dass  $M$  ausgehend von der Konfiguration  $(q, \varepsilon, A)$  unendlich viele  $\varepsilon$ -Übergänge ausführt und dabei auch Endzustände besucht.  
 $e\varepsilon A \rightarrow fA$ ,

- Schließlich übernehmen wir von  $M$  noch

(f) alle Anweisungen aus  $\delta$ , soweit sie nicht durch Anweisungen vom Typ (d) oder (e) überschrieben wurden.

# DPDAs, die ihre Eingabe zu Ende lesen

## Beweis (Schluss)

Um dies zu verhindern, transformieren wir  $M = (Z, \Sigma, \Gamma, \delta, q_0, \#, E)$  in den DPDA

$$M' = (Z \cup \{s, e, f\}, \Sigma, \Gamma', \delta', s, \#, E \cup \{e\}) \text{ mit } \Gamma' = \Gamma \cup \{\square\},$$

wobei  $\delta'$  folgende Anweisungen enthält:

- (a)  $s\varepsilon\# \rightarrow q_0\#\square$ ,
- (b)  $qaA \rightarrow fA$ , für alle  $(q, a, A) \in Z \times \Sigma \times \Gamma'$  mit  $A = \square$  oder  $\delta(q, a, A) = \delta(q, \varepsilon, A) = \emptyset$ ,
- (c)  $faA \rightarrow fA$ , für alle  $a \in \Sigma$  und  $A \in \Gamma'$ ,
- (d)  $q\varepsilon A \rightarrow fA$ , für alle  $q \in Z$  und  $A \in \Gamma$ , so dass ausgehend von der Konfiguration  $(q, \varepsilon, A)$  unendlich viele  $\varepsilon$ -Übergänge ausgeführt werden, ohne dass dabei ein Endzustand besucht wird.
- (e)  $q\varepsilon A \rightarrow eA$   
 $e\varepsilon A \rightarrow fA$ , für alle  $q \in Z$  und  $A \in \Gamma$ , so dass ausgehend von der Konfiguration  $(q, \varepsilon, A)$  unendlich viele  $\varepsilon$ -Übergänge ausgeführt und dabei auch Endzustände besucht werden,
- (f) alle Anweisungen aus  $\delta$ , soweit sie nicht durch Anweisungen vom Typ (c) oder (e) überschrieben wurden.

# DPDAs, die ihre Eingabe zu Ende lesen

## Beispiel

Wenden wir diese Konstruktion auf den DPDA

$$M = (\{q_0, q_1, q_2\}, \{a, b, c\}, \{A, B, \#\}, \delta, q_0, \#, \{q_2\})$$

mit der Überföhrungsfunktion

$\delta$	$q_0, \#$	$q_0, A$	$q_0, B$	$q_1, \#$	$q_1, A$	$q_1, B$	$q_2, \#$	$q_2, A$	$q_2, B$
$\varepsilon$	—	—	—	$q_2$	—	—	$q_2\#$	—	—
$a$	$q_0A\#$	$q_0AA$	$q_0AB$	—	$q_1$	—	—	—	—
$b$	$q_0B\#$	$q_0BA$	$q_0BB$	—	—	$q_1$	—	—	—
$c$	—	$q_1A$	$q_1B$	—	—	—	—	—	—

an, so erhalten wir den DPDA

$$M' = (\{q_0, q_1, q_2, s, e, f\}, \{a, b, c\}, \{A, B, \#, \square\}, \delta', s, \#, \{q_2, e\})$$

mit folgender Überföhrungsfunktion  $\delta'$ :

# DPDAs, die ihre Eingabe zu Ende lesen

## Beispiel (Schluss)

$\delta'$	$q_0, \#$	$q_0, A$	$q_0, B$	$q_0, \square$	$q_1, \#$	$q_1, A$	$q_1, B$	$q_1, \square$	$q_2, \#$	$q_2, A$	$q_2, B$	$q_2, \square$
$\varepsilon$	—	—	—	—	$q_2$	—	—	—	$e\#$	—	—	—
$a$	$q_0A\#$	$q_0AA$	$q_0AB$	$f\square$	—	$q_1$	$fB$	$f\square$	—	$fA$	$fB$	$f\square$
$b$	$q_0B\#$	$q_0BA$	$q_0BB$	$f\square$	—	$fA$	$q_1$	$f\square$	—	$fA$	$fB$	$f\square$
$c$	$f\#$	$q_1A$	$q_1B$	$f\square$	—	$fA$	$fB$	$f\square$	—	$fA$	$fB$	$f\square$
<i>Typ</i>	$(f, b)$	$(f)$	$(f)$	$(b)$	$(f)$	$(f, b)$	$(f, b)$	$(b)$	$(e)$	$(b)$	$(b)$	$(b)$

	$s, \#$	$s, A$	$s, B$	$s, \square$	$f, \#$	$f, A$	$f, B$	$f, \square$	$e, \#$	$e, A$	$e, B$	$e, \square$
$\varepsilon$	$q_0\#\square$	—	—	—	—	—	—	—	$f\#$	—	—	—
$a$	—	—	—	—	$f\#$	$fA$	$fB$	$f\square$	—	—	—	—
$b$	—	—	—	—	$f\#$	$fA$	$fB$	$f\square$	—	—	—	—
$c$	—	—	—	—	$f\#$	$fA$	$fB$	$f\square$	—	—	—	—
<i>Typ</i>	$(a)$				$(c)$	$(c)$	$(c)$	$(c)$	$(e)$			

# Komplementabschluss von DCFL

## Satz

Die Klasse DCFL ist unter Komplement abgeschlossen.

## Beweis

- Sei  $M = (Z, \Sigma, \Gamma, \delta, q_0, \#, E)$  ein DPDA, der alle Eingaben zu Ende liest, und sei  $L(M) = L$ .
- Wir konstruieren einen DPDA  $\bar{M}$  für  $\bar{L}$ , der  $M$  simuliert.
- Dabei merkt sich  $\bar{M}$  in seinem Zustand  $(q, i)$  neben dem aktuellen Zustand  $q$  von  $M$  in der Komponente  $i$ , ob  $M$  nach Lesen des letzten Zeichens (bzw. seit Rechnungsbeginn) einen Endzustand besucht hat ( $i = 2$ ) oder nicht ( $i = 1$ ).
- Möchte  $M$  das nächste Zeichen lesen und befindet sich  $\bar{M}$  im Zustand  $(q, 1)$ , so macht  $\bar{M}$  noch einen Umweg über den Endzustand  $(q, 3)$ .

# Komplementabschluss von DCFL

## Beweis (Schluss)

- Konkret sei  $\overline{M} = (Z \times \{1, 2, 3\}, \Sigma, \Gamma, \delta', s, \#, Z \times \{3\})$  mit

$$s = \begin{cases} (q_0, 1), & q_0 \notin E, \\ (q_0, 2), & \text{sonst,} \end{cases}$$

wobei  $\delta'$  für jede Anweisung  $q \in A \rightarrow_M p \gamma$  die Anweisungen

$$\begin{aligned} (q, 1) \in A &\rightarrow (p, 1) \gamma, & \text{falls } p \notin E, \\ (q, 1) \in A &\rightarrow (p, 2) \gamma, & \text{falls } p \in E \text{ und} \\ (q, 2) \in A &\rightarrow (p, 2) \gamma, \end{aligned}$$

sowie für jede Anweisung  $qaA \rightarrow_M p \gamma$  folgende Anweisungen enthält:

$$\begin{aligned} (q, 1) \in A &\rightarrow (q, 3) A, \\ (q, 2) aA &\rightarrow (p, 1) \gamma, & \text{falls } p \notin E, \\ (q, 2) aA &\rightarrow (p, 2) \gamma, & \text{falls } p \in E, \\ (q, 3) aA &\rightarrow (p, 1) \gamma, & \text{falls } p \notin E \text{ und} \\ (q, 3) aA &\rightarrow (p, 2) \gamma, & \text{falls } p \in E. \end{aligned}$$

Man beachte, dass  $\overline{M}$  in einem Endzustand keine  $\varepsilon$ -Übergänge macht. □

# Komplementabschluss von DCFL

## Beispiel

- Angenommen, ein DPDA  $M = (Z, \Sigma, \Gamma, \delta, q_0, \#, E)$  führt bei Eingabe  $x = a$  folgende Rechnung aus:

$$(q_0, a, \#) \vdash (q_1, \varepsilon, \gamma_1) \vdash (q_2, \varepsilon, \gamma_2).$$

- Dann würde  $\bar{M}$  im Fall  $E = \{q_0, q_2\}$  (d.h.  $x \in L(M)$ ) die Rechnung

$$((q_0, 2), a, \#) \vdash ((q_1, 1), \varepsilon, \gamma_1) \vdash ((q_2, 2), \varepsilon, \gamma_2)$$

ausführen und das Wort  $a$  verwerfen, da  $(q_1, 1), (q_2, 2) \notin Z \times \{3\}$  sind.

- Im Fall  $E = \{q_0\}$  (d.h.  $x \notin L(M)$ ) würde  $\bar{M}$  dagegen die Rechnung

$$((q_0, 2), a, \#) \vdash ((q_1, 1), \varepsilon, \gamma_1) \vdash ((q_2, 1), \varepsilon, \gamma_2) \vdash ((q_2, 3), \varepsilon, \gamma_2)$$

ausführen und das Wort  $a$  akzeptieren, da  $(q_2, 3) \in Z \times \{3\}$  ist.

# Komplementabschluss von DCFL

## Definition

Für eine Sprachklasse  $\mathcal{C}$  bezeichne  $\text{co-}\mathcal{C}$  die Klasse  $\{\bar{L} \mid L \in \mathcal{C}\}$  aller Komplemente von Sprachen in  $\mathcal{C}$ .

## Korollar

- $\text{REG} = \text{co-REG}$ ,
- $\text{DCFL} = \text{co-DCFL}$ ,
- $\text{CFL} \neq \text{co-CFL}$ .



## Weitere Abschlusseigenschaften von DCFL

### Satz

Die Klasse DCFL ist nicht abgeschlossen unter Durchschnitt, Vereinigung, Produkt und Sternhülle.

Beweis von  $L_1, L_2 \in \text{DCFL} \not\Rightarrow L_1 \cap L_2 \in \text{DCFL}$

- Die beiden Sprachen

$$L_1 = \{a^n b^m c^m \mid n, m \geq 0\} \quad \text{und} \quad L_2 = \{a^n b^n c^m \mid n, m \geq 0\}$$

sind sogar deterministisch kontextfrei (siehe Übungen).

- Da  $L_1 \cap L_2 = \{a^n b^n c^n \mid n \geq 0\}$  nicht kontextfrei ist, liegt der Schnitt dieser Sprachen natürlich auch nicht in DCFL.

## Beweis von $L_1, L_2 \in \text{DCFL} \not\Rightarrow L_1 \cup L_2 \in \text{DCFL}$

- Da DCFL unter Komplementbildung abgeschlossen ist, kann DCFL wegen de Morgan dann auch nicht unter Vereinigung abgeschlossen sein.

- Beispielsweise sind folgende Sprachen deterministisch kontextfrei:

$$L_3 = \{a^i b^j c^k \mid i \neq j\} \text{ und } L_4 = \{a^i b^j c^k \mid j \neq k\}.$$

- Ihre Vereinigung gehört aber nicht zu DCFL, d.h.

$$L_3 \cup L_4 = \{a^i b^j c^k \mid i \neq j \text{ oder } j \neq k\} \in \text{CFL} \setminus \text{DCFL}.$$

- DCFL ist nämlich unter Schnitt mit regulären Sprachen abgeschlossen (siehe Übungen).
- Daher wäre mit  $L_3 \cup L_4$  auch die Sprache

$$\overline{(L_3 \cup L_4)} \cap L(a^* b^* c^*) = \{a^n b^n c^n \mid n \geq 0\}$$

(deterministisch) kontextfrei.

## Beweis von $L_1, L_2 \in \text{DCFL} \not\Rightarrow L_1L_2 \in \text{DCFL}$

- Betrachte die DCFL Sprachen

$$L_0 = \{0\}, L_3 = \{a^i b^j c^k \mid i \neq j\} \text{ und } L_4 = \{a^i b^j c^k \mid j \neq k\}.$$

- Wir wissen bereits, dass  $L_3 \cup L_4 \notin \text{DCFL}$  ist.
- Dann ist aber auch die Sprache

$$L_5 = L_0(L_3 \cup L_4) = \{0a^i b^j c^k \mid i \neq j \vee j \neq k\} \notin \text{DCFL},$$

da sich ein DPDA  $M = (Z, \Sigma, \Gamma, \delta, q_0, \#, E)$  für  $L_5$  leicht zu einem DPDA für  $L_3 \cup L_4$  umbauen ließe:

- Sei  $(p, \varepsilon, \gamma)$  die Konfiguration, die  $M$  nach Lesen der Eingabe 0 erreicht.
- Dann erkennt der DPDA  $M' = (Z \cup \{s\}, \Sigma, \Gamma, \delta', s, \#, E)$  die Sprache  $L_3 \cup L_4$ , wobei  $\delta'$  wie folgt definiert ist:

$$\delta'(q, u, A) = \begin{cases} (p, \gamma), & (q, u, A) = (s, \varepsilon, \#), \\ \delta(q, u, A), & (q, u, A) \in Z \times (\Sigma \cup \{\varepsilon\}) \times \Gamma. \end{cases}$$

## Beweis von $L_1, L_2 \in \text{DCFL} \not\Rightarrow L_1L_2 \in \text{DCFL}$

- Betrachte die DCFL Sprachen

$$L_0 = \{0\}, L_3 = \{a^i b^j c^k \mid i \neq j\} \text{ und } L_4 = \{a^i b^j c^k \mid j \neq k\}.$$

- Es ist leicht zu sehen, dass auch die beiden Sprachen  $L_0^*$  und  $L = L_0L_3 \cup L_4$  in DCFL sind (siehe Übungen).
- Ihr Produkt  $L_0^*L$  gehört aber nicht zu DCFL.
- Da DCFL unter Schnitt mit regulären Sprachen abgeschlossen ist, wäre andernfalls auch die Sprache

$$L_0^*L \cap L_0L(a^*b^*c^*) = \{0a^i b^j c^k \mid i \neq j \vee j \neq k\} = L_0(L_3 \cup L_4) = L_5$$

in DCFL, was wir bereits ausgeschlossen haben.

### Bemerkung

Dass DCFL auch nicht unter Sternhüllenbildung abgeschlossen ist, lässt sich ganz ähnlich zeigen (siehe Übungen).

# Abschlusseigenschaften

## Abschlusseigenschaften der Klassen REG, DCFL und CFL

	Vereinigung	Schnitt	Komplement	Produkt	Sternhülle
REG	<i>ja</i>	<i>ja</i>	<i>ja</i>	<i>ja</i>	<i>ja</i>
DCFL	<i>nein</i>	<i>nein</i>	<i>ja</i>	<i>nein</i>	<i>nein</i>
CFL	<i>ja</i>	<i>nein</i>	<i>nein</i>	<i>ja</i>	<i>ja</i>

# Die Chomsky-Hierarchie

## Definition

Sei  $G = (V, \Sigma, P, S)$  eine Grammatik.

- ①  $G$  heißt vom Typ 3 oder regulär, falls für alle Regeln  $u \rightarrow v$  gilt:

$$u \in V \text{ und } v \in \Sigma V \cup \Sigma \cup \{\varepsilon\}.$$

(d.h. alle Regeln haben die Form  $A \rightarrow aB$ ,  $A \rightarrow a$  oder  $A \rightarrow \varepsilon$ )

- ②  $G$  heißt vom Typ 2 oder kontextfrei, falls für alle Regeln  $u \rightarrow v$  gilt:

$$u \in V. \quad (\text{d.h. alle Regeln haben die Form } A \rightarrow \alpha)$$

- ③  $G$  heißt vom Typ 1 oder kontextsensitiv, falls für alle Regeln  $u \rightarrow v$  gilt:

$$|v| \geq |u|. \quad (\text{mit Ausnahme der } \varepsilon\text{-Sonderregel, s. unten})$$

- ④ Jede Grammatik ist automatisch vom Typ 0.

## Die $\varepsilon$ -Sonderregel

In einer kontextsensitiven Grammatik ist auch die Regel  $S \rightarrow \varepsilon$  zulässig. Aber nur, wenn das Startsymbol  $S$  nur links vorkommt.

## CFL ist echt in CSL enthalten

### Bemerkung

- Wie wir gesehen haben, ist CFL in CSL enthalten.
- Zudem ist folgende Sprache nicht kontextfrei:

$$L = \{a^n b^n c^n \mid n \geq 1\}.$$

- $L$  kann jedoch von einer kontextsensitiven Grammatik erzeugt werden.
- Daher ist CFL echt in CSL enthalten.

# Eine kontextsensitive Grammatik für $\{a^n b^n c^n \mid n \geq 1\}$

## Beispiel

- Betrachte die Grammatik  $G = (V, \Sigma, P, S)$  mit  $V = \{S, B, C\}$ ,  $\Sigma = \{a, b, c\}$  und den Regeln

$$P: \quad S \rightarrow aSBC, aBC \quad (1, 2) \quad CB \rightarrow BC \quad (3) \quad aB \rightarrow ab \quad (4) \\ bB \rightarrow bb \quad (5) \quad C \rightarrow c \quad (6)$$

- In  $G$  lässt sich beispielsweise das Wort  $w = aabbcc$  ableiten:

$$S \xRightarrow{(1)} aSBC \xRightarrow{(2)} aaBCBC \xRightarrow{(3)} aaBBCC \\ \xRightarrow{(4)} aabBCC \xRightarrow{(5)} aabbCC \xRightarrow{(6)} aabbcC \xRightarrow{(6)} aabbcc$$

- Allgemein gilt für alle  $n \geq 1$ :

$$S \xRightarrow{(1)} a^{n-1} S(BC)^{n-1} \xRightarrow{(2)} a^n (BC)^n \xRightarrow{(3)} a^n B^n C^n \\ \xRightarrow{(4)} a^n b B^{n-1} C^n \xRightarrow{(5)} a^n b^n C^n \xRightarrow{(6)} a^n b^n c^n$$

- Also gilt  $a^n b^n c^n \in L(G)$  für alle  $n \geq 1$ .

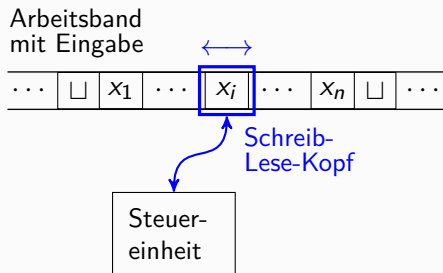


# Eine kontextsensitive Grammatik für $\{a^n b^n c^n \mid n \geq 1\}$

## Beispiel (Schluss)

- Betrachte die kontextsensitive Grammatik  $G = (V, \Sigma, P, S)$  mit  $V = \{S, B, C\}$ ,  $\Sigma = \{a, b, c\}$  und den Regeln
$$P: \quad S \rightarrow aSBC, aBC \quad (1,2) \quad CB \rightarrow BC \quad (3) \quad aB \rightarrow ab \quad (4)$$
$$bB \rightarrow bb \quad (5) \quad C \rightarrow c \quad (6)$$
- Umgekehrt folgt durch Induktion über die Ableitungslänge, dass jede Satzform  $u$  mit  $S \Rightarrow^* u$  die folgenden Bedingungen erfüllt:
  - $\#_a(u) = \#_b(u) + \#_B(u) = \#_c(u) + \#_C(u)$ ,
  - links von  $S$  kommen nur  $a$ 's vor,
  - links von einem  $a$  kommen ebenfalls nur  $a$ 's vor,
  - links von einem  $b$  kommen nur  $a$ 's oder  $b$ 's vor.
- Daraus ergibt sich, dass in  $G$  nur Wörter  $w \in \Sigma^*$  der Form  $w = a^n b^n c^n$  ableitbar sind, d.h.  $L(G) = \{a^n b^n c^n \mid n \geq 1\} \in \text{CSL}$ .

# Die Turingmaschine



- Um ein geeignetes Maschinenmodell für die kontextsensitiven Sprachen zu finden, führen wir zunächst das Rechenmodell der nichtdeterministischen Turingmaschine (NTM) ein.
- Eine NTM erhält ihre Eingabe auf einem nach links und rechts unbegrenzten Band.
- Während ihrer Rechnung kann sie den Schreib-Lese-Kopf auf dem Band in beide Richtungen bewegen und dabei die besuchten Bandfelder lesen sowie die gelesenen Zeichen gegebenenfalls überschreiben.

# Das Rechenmodell der Turingmaschine

## Definition

- Sei  $k \geq 1$ . Eine **nichtdeterministische  $k$ -Band-Turingmaschine** ( **$k$ -NTM** oder einfach **NTM**) wird durch ein 6-Tupel  $M = (Z, \Sigma, \Gamma, \delta, q_0, E)$  beschrieben, wobei
  - $Z$  eine endliche Menge von Zuständen,
  - $\Sigma$  das Eingabealphabet (mit  $\sqcup \notin \Sigma$ ),
  - $\Gamma$  das Arbeitsalphabet (mit  $\Sigma \cup \{\sqcup\} \subseteq \Gamma$ ),
  - $\delta: Z \times \Gamma^k \rightarrow \mathcal{P}(Z \times \Gamma^k \times \{L, R, N\}^k)$  die Überföhrungsfunktion,
  - $q_0$  der Startzustand und
  - $E \subseteq Z$  die Menge der Endzustände ist.
- Eine  $k$ -NTM  $M$  heißt **deterministisch** (kurz:  $M$  ist eine  **$k$ -DTM** oder einfach **DTM**), falls für alle  $(q, a_1, \dots, a_k) \in Z \times \Gamma^k$  gilt:

$$\|\delta(q, a_1, \dots, a_k)\| \leq 1.$$

# Das Rechenmodell der Turingmaschine

- Für  $(q, b_1, \dots, b_k, D_1, \dots, D_k) \in \delta(p, a_1, \dots, a_k)$  schreiben wir auch  $(p, a_1, \dots, a_k) \rightarrow (q, b_1, \dots, b_k, D_1, \dots, D_k)$ .
- Eine solche **Anweisung** ist ausführbar, falls
  - $p$  der aktuelle Zustand von  $M$  ist und
  - sich für  $i = 1, \dots, k$  der Lesekopf des  $i$ -ten Bandes auf einem mit  $a_i$  beschrifteten Feld befindet.
- Ihre Ausführung bewirkt, dass  $M$ 
  - vom Zustand  $p$  in den Zustand  $q$  übergeht,
  - auf Band  $i$  das Symbol  $a_i$  durch  $b_i$  ersetzt und
  - den Kopf gemäß  $D_i$  bewegt (L: ein Feld nach links, R: ein Feld nach rechts, N: keine Bewegung).

# Das Rechenmodell der Turingmaschine

## Definition

- Eine **Konfiguration** ist ein  $(3k + 1)$ -Tupel

$$K = (q, u_1, a_1, v_1, \dots, u_k, a_k, v_k) \in Z \times (\Gamma^* \times \Gamma \times \Gamma^*)^k$$

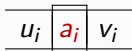
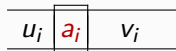
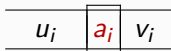
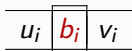
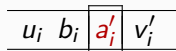
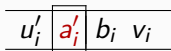
und besagt, dass

- $q$  der momentane Zustand ist und
  - das  $i$ -te Band mit  $\dots \sqcup u_i a_i v_i \sqcup \dots$  beschriftet ist, wobei sich der Kopf auf dem Zeichen  $a_i$  befindet.
- Im Fall  $k = 1$  schreiben wir für eine Konfiguration  $(q, u, a, v)$  auch kurz  $uqav$ .

# Das Rechenmodell der Turingmaschine

## Definition

Seien  $K = (p, u_1, a_1, v_1, \dots, u_k, a_k, v_k)$  und  $K' = (q, u'_1, a'_1, v'_1, \dots, u'_k, a'_k, v'_k)$  Konfigurationen.  $K'$  heißt **Folgekonfiguration** von  $K$  (kurz  $K \vdash K'$ ), falls eine Anweisung  $(p, a_1, \dots, a_k) \rightarrow (q, b_1, \dots, b_k, D_1, \dots, D_k)$  existiert, so dass für  $i = 1, \dots, k$  gilt:

im Fall $D_i = N$ :	$D_i = R$ :	$D_i = L$ :
$K$ : 	$K$ : 	$K$ : 
$K'$ : 	$K'$ : 	$K'$ : 
$u'_i = u_i,$ $a'_i = b_i$ und $v'_i = v_i.$	$u'_i = u_i b_i$ und $a'_i v'_i = \begin{cases} v_i, & v_i \neq \varepsilon, \\ \sqcup, & \text{sonst.} \end{cases}$	$u'_i a'_i = \begin{cases} u_i, & u_i \neq \varepsilon, \\ \sqcup, & \text{sonst} \end{cases}$ und $v'_i = b_i v_i.$

# Das Rechenmodell der Turingmaschine

## Definition

- Die **Startkonfiguration** von  $M$  bei Eingabe  $x = x_1 \cdots x_n \in \Sigma^*$  ist

$$K_x = \begin{cases} (q_0, \varepsilon, x_1, x_2 \cdots x_n, \varepsilon, \sqcup, \varepsilon, \dots, \varepsilon, \sqcup, \varepsilon), & x \neq \varepsilon, \\ (q_0, \varepsilon, \sqcup, \varepsilon, \dots, \varepsilon, \sqcup, \varepsilon), & x = \varepsilon. \end{cases}$$

- Eine **Rechnung** von  $M$  bei Eingabe  $x$  ist eine (endliche oder unendliche) Folge von Konfigurationen  $K_0, K_1, K_2 \dots$  mit  $K_0 = K_x$  und  $K_0 \vdash K_1 \vdash K_2 \dots$ .
- Die von  $M$  **akzeptierte** oder **erkannte Sprache** ist

$$L(M) = \{x \in \Sigma^* \mid \exists K \in E \times (\Gamma^* \times \Gamma \times \Gamma^*)^k : K_x \vdash^* K\}.$$

- Ein Wort  $x$  wird also genau dann von  $M$  akzeptiert (kurz:  **$M(x)$  akzeptiert**), wenn es eine Rechnung von  $M$  bei Eingabe  $x$  gibt, bei der ein Endzustand erreicht wird.

# Das Rechenmodell der Turingmaschine

## Beispiel

Betrachte die 1-DTM  $M = (Z, \Sigma, \Gamma, \delta, q_0, E)$  mit  $Z = \{q_0, \dots, q_4\}$ ,  $\Sigma = \{a, b\}$ ,  $\Gamma = \Sigma \cup \{A, B, \sqcup\}$ ,  $E = \{q_4\}$  und den Anweisungen

- $\delta: q_0 a \rightarrow q_1 AR$  (1) Anfang der Schleife: Ersetze das erste  $a$  durch  $A$
- $q_1 a \rightarrow q_1 aR$  (2) Bewege den Kopf nach rechts bis zum ersten  $b$
- $q_1 B \rightarrow q_1 BR$  (3) und ersetze dies durch ein  $B$  (falls kein  $b$  mehr
- $q_1 b \rightarrow q_2 BL$  (4) vorhanden ist, dann halte ohne zu akzeptieren).
- $q_2 a \rightarrow q_2 aL$  (5) Bewege den Kopf nach links bis ein  $A$  kommt,
- $q_2 B \rightarrow q_2 BL$  (6) gehe ein Feld nach rechts zurück und wiederhole
- $q_2 A \rightarrow q_0 AR$  (7) die Schleife.
- $q_0 B \rightarrow q_3 BR$  (8) Falls kein  $a$  am Anfang der Schleife, dann teste,
- $q_3 B \rightarrow q_3 BR$  (9) ob noch ein  $b$  vorhanden ist. Wenn ja, dann halte
- $q_3 \sqcup \rightarrow q_4 \sqcup N$  (10) ohne zu akzeptieren. Andernfalls akzeptiere.





# Das Rechenmodell der Turingmaschine

## Beispiel (Schluss)

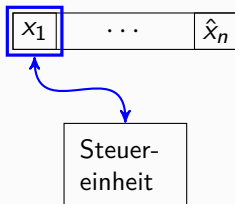
$$\begin{aligned} \delta: q_0 a &\rightarrow q_1 AR & (1) & \quad q_1 a &\rightarrow q_1 aR & (2) & \quad q_1 B &\rightarrow q_1 BR & (3) \\ q_1 b &\rightarrow q_2 BL & (4) & \quad q_2 a &\rightarrow q_2 aL & (5) & \quad q_2 B &\rightarrow q_2 BL & (6) \\ q_2 A &\rightarrow q_0 AR & (7) & \quad q_0 B &\rightarrow q_3 BR & (8) & \quad q_3 B &\rightarrow q_3 BR & (9) \\ q_3 \sqcup &\rightarrow q_4 \sqcup N & (10) \end{aligned}$$

- Andererseits führt die Eingabe  $abb$  auf die Rechnung

$$q_0 abb \underset{(1)}{\vdash} Aq_1 bb \underset{(4)}{\vdash} q_2 ABb \underset{(7)}{\vdash} Aq_0 Bb \underset{(8)}{\vdash} ABq_3 b$$

- Da diese nicht fortsetzbar ist und da  $M$  deterministisch ist, kann  $M(abb)$  nicht den Endzustand  $q_4$  erreichen, d.h.  $abb$  gehört nicht zu  $L(M)$ .
- Tatsächlich lässt sich zeigen, dass  $L(M) = \{a^n b^n \mid n \geq 1\}$  ist.
- In den Übungen werden wir eine 1-DTM für die Sprache  $L = \{a^n b^n c^n \mid n \geq 1\}$  konstruieren.

## Ein Maschinenmodell für CSL



- Es ist leicht zu sehen, dass jede Typ-0 Sprache von einer NTM  $M$  erkannt wird, die ausgehend von der Eingabe  $x$  eine Rückwärtsableitung (Reduktion) auf das Startsymbol sucht.
- Im Fall einer Typ-1 Sprache ist die linke Seite jeder Regel höchstens so lang wie die rechte Seite.
- Daher muss  $M$  in diesem Fall nur deshalb das Blank hinter  $x$  lesen, um das Ende der Eingabe erkennen zu können.
- Falls wir das letzte Zeichen  $x_n$  von  $x$  markieren, kann  $M$  jedoch die Rechnung auf den Bereich der Eingabe beschränken.
- NTMs mit dieser Eigenschaft werden auch als LBAs bezeichnet.

# Linear beschränkte Automaten

## Definition

- Für ein Alphabet  $\Sigma$  sei  $\hat{\Sigma} = \Sigma \cup \{\hat{a} \mid a \in \Sigma\}$  und für  $x = x_1 \cdots x_n \in \Sigma^*$

sei

$$\hat{x} = \begin{cases} x, & x = \varepsilon, \\ x_1 \cdots x_{n-1} \hat{x}_n, & x \neq \varepsilon. \end{cases}$$

- Eine 1-NTM  $M = (Z, \hat{\Sigma}, \Gamma, \delta, q_0, E)$  heißt **linear beschränkt** (kurz:  $M$  ist ein **LBA**), falls gilt:

$$\forall x \in \Sigma^* : K_{\hat{x}} \vdash^* uqav \Rightarrow |uav| \leq \max\{|x|, 1\}.$$

- Die von einem LBA **akzeptierte** oder **erkannte Sprache** ist

$$L(M) = \{x \in \Sigma^* \mid M(\hat{x}) \text{ akzeptiert}\}.$$

## Bemerkung

Jede  $k$ -NTM, die bei Eingaben der Länge  $n$  höchstens linear viele (also  $cn + c$  für eine Konstante  $c$ ) Bandfelder besucht, kann von einem LBA simuliert werden.

# Linear beschränkte Automaten

## Beispiel

- Es ist nicht schwer, die 1-DTM  $M = (Z, \Sigma, \Gamma, \delta, q_0, E)$  mit

$$\delta: q_0 a \rightarrow q_1 AR \quad (1) \quad q_1 a \rightarrow q_1 aR \quad (2) \quad q_1 B \rightarrow q_1 BR \quad (3)$$

$$q_1 b \rightarrow q_2 BL \quad (4) \quad q_2 a \rightarrow q_2 aL \quad (5) \quad q_2 B \rightarrow q_2 BL \quad (6)$$

$$q_2 A \rightarrow q_0 AR \quad (7) \quad q_0 B \rightarrow q_3 BR \quad (8) \quad q_3 B \rightarrow q_3 BR \quad (9)$$

$$q_3 \sqcup \rightarrow q_4 \sqcup N \quad (10)$$

in einen deterministischen LBA (kurz: **DLBA**)  $M' = (Z, \hat{\Sigma}, \Gamma', \delta', q_0, E)$  für die Sprache  $\{a^n b^n \mid n \geq 1\}$  umzuwandeln.

- Ersetze hierzu

- $\Sigma$  durch  $\hat{\Sigma} = \{a, b, \hat{a}, \hat{b}\}$ ,

- $\Gamma$  durch  $\Gamma' = \hat{\Sigma} \cup \{A, B, \hat{B}, \sqcup\}$  sowie

- die Anweisung  $q_3 \sqcup \rightarrow q_4 \sqcup N$  (10) durch  $q_3 \hat{B} \rightarrow q_4 \hat{B} N$  (10')

und füge die Anweisungen  $q_1 \hat{b} \rightarrow q_2 \hat{B} L$  (4a) und  $q_0 \hat{B} \rightarrow q_4 \hat{B} N$  (8a) hinzu.

# Linear beschränkte Automaten

## Beispiel

- Ersetze hierzu
  - $\Sigma$  durch  $\hat{\Sigma} = \{a, b, \hat{a}, \hat{b}\}$ ,
  - $\Gamma$  durch  $\Gamma' = \hat{\Sigma} \cup \{A, B, \hat{B}, \sqcup\}$  sowie
  - die Anweisung  $q_3\sqcup \rightarrow q_4\sqcup N$  (10) durch  $q_3\hat{B} \rightarrow q_4\hat{B}N$  (10')und füge die Anweisungen  $q_1\hat{b} \rightarrow q_2\hat{B}L$  (4a) und  $q_0\hat{B} \rightarrow q_4\hat{B}N$  (8a) hinzu:

$$\begin{array}{llll} \delta': q_0a \rightarrow q_1AR & (1) & q_1a \rightarrow q_1aR & (2) & q_1B \rightarrow q_1BR & (3) \\ q_1b \rightarrow q_2BL & (4) & q_1\hat{b} \rightarrow q_2\hat{B}L & (4a) & q_2a \rightarrow q_2aL & (5) \\ q_2B \rightarrow q_2BL & (6) & q_2A \rightarrow q_0AR & (7) & q_0B \rightarrow q_3BR & (8) \\ q_0\hat{B} \rightarrow q_4\hat{B}N & (8a) & q_3B \rightarrow q_3BR & (9) & q_3\hat{B} \rightarrow q_4\hat{B}N & (10') \end{array}$$

- Dann akzeptiert  $M'$  die Eingabe  $aab\hat{b}$  wie folgt (d.h.  $aabb \in L(M')$ ):

$$q_0aab\hat{b} \vdash^* AABq_1\hat{b} \underset{(4a)}{\vdash} AAq_2B\hat{B} \vdash^* AABq_3\hat{B} \underset{(10')}{\vdash} AABq_4\hat{B}$$

# Charakterisierung von CSL mittels LBAs

Als nächstes zeigen wir, dass LBAs genau die kontextsensitiven Sprachen erkennen.

## Satz

$CSL = \{L(M) \mid M \text{ ist ein LBA}\}.$

## Beweis von $CSL \subseteq \{L(M) \mid M \text{ ist ein LBA}\}$

Sei  $G = (V, \Sigma, P, S)$  eine kontextsensitive Grammatik. Dann wird  $L(G)$  von folgendem LBA  $M$  akzeptiert (o.B.d.A. sei  $\varepsilon \notin L(G)$ ):

Arbeitsweise von  $M$  bei Eingabe  $x = x_1 \cdots x_n$  mit  $n > 0$ :

- 1 Markiere das erste Eingabezeichen  $x_1$
- 2 Wähle (nichtdeterministisch) eine Regel  $\alpha \rightarrow \beta$  aus  $P$
- 3 Wähle ein beliebiges Vorkommen von  $\beta$  auf dem Band  
(falls  $\beta$  nicht vorkommt, halte ohne zu akzeptieren)
- 4 Ersetze die ersten  $|\alpha|$  Zeichen von  $\beta$  durch  $\alpha$
- 5 Falls das erste (oder letzte) Zeichen von  $\beta$  markiert war, markiere auch das erste (letzte) Zeichen von  $\alpha$
- 6 Verschiebe die Zeichen rechts von  $\beta$  um  $|\beta| - |\alpha|$  Positionen nach links und überschreibe die frei werdenden Bandfelder mit Blanks
- 7 Enthält das Band außer Blanks nur das (markierte) Startsymbol, so halte in einem Endzustand
- 8 Gehe zurück zu Schritt 2



## Beweis von $CSL \subseteq \{L(M) \mid M \text{ ist ein LBA}\}$

- Nun ist leicht zu sehen, dass  $M$  wegen  $|\beta| \geq |\alpha|$  tatsächlich ein LBA ist.
- $M$  akzeptiert  $x$ , falls es gelingt, eine Ableitung für  $x$  in  $G$  zu finden (in umgekehrter Reihenfolge, d.h.  $M$  ist ein nichtdeterministischer *Bottom-Up Parser*).
- Da sich genau für die Wörter in  $L(G)$  eine Ableitung finden lässt, folgt  $L(M) = L(G)$ . □

## Beweis von $\{L(M) \mid M \text{ ist ein LBA}\} \subseteq \text{CSL}$

- Sei  $M = (Z, \hat{\Sigma}, \Gamma, \delta, q_0, E)$  ein LBA (o.B.d.A. sei  $\varepsilon \notin L(M)$ ).
- Betrachte die kontextsensitive Grammatik  $G = (V, \Sigma, P, S)$  mit

$$V = \{S, A\} \cup (Z\Gamma \cup \Gamma) \times \Sigma,$$

die für alle  $a, b \in \Sigma$  und  $c, d \in \Gamma$  folgende Regeln enthält:

$P:$	$S \rightarrow A(\hat{a}, a), (q_0\hat{a}, a)$	(S)	„Startregeln“
	$A \rightarrow A(a, a), (q_0a, a)$	(A)	„A-Regeln“
	$(c, a) \rightarrow a$	(F)	„Finale Regeln“
	$(qc, a) \rightarrow a,$	falls $q \in E$	(E) „E-Regeln“
	$(qc, a) \rightarrow (q'c', a),$	falls $qc \rightarrow_M q'c'N$	(N) „N-Regeln“
	$(qc, a)(d, b) \rightarrow (c', a)(q'd, b),$	falls $qc \rightarrow_M q'c'R$	(R) „R-Regeln“
	$(d, a)(qc, b) \rightarrow (q'd, a)(c', b),$	falls $qc \rightarrow_M q'c'L$	(L) „L-Regeln“

# Beweis von $\{L(M) \mid M \text{ ist ein LBA}\} \subseteq \text{CSL}$

## Beispiel

- Betrachte den LBA  $M = (Z, \hat{\Sigma}, \Gamma, \delta, q_0, E)$  mit  $Z = \{q_0, \dots, q_4\}$ ,  $\Sigma = \{a, b\}$ ,  $\Gamma = \{a, b, \hat{a}, \hat{b}, A, B, \hat{B}, \sqcup\}$  und  $E = \{q_4\}$ , sowie

$$\begin{array}{lll} \delta: q_0 a \rightarrow q_1 A R & q_1 \hat{b} \rightarrow q_2 \hat{B} L & q_0 B \rightarrow q_3 B R \\ q_1 a \rightarrow q_1 a R & q_2 a \rightarrow q_2 a L & q_0 \hat{B} \rightarrow q_4 \hat{B} N \\ q_1 B \rightarrow q_1 B R & q_2 B \rightarrow q_2 B L & q_3 B \rightarrow q_3 B R \\ q_1 b \rightarrow q_2 B L & q_2 A \rightarrow q_0 A R & q_3 \hat{B} \rightarrow q_4 \hat{B} N \end{array}$$

- Die zugehörige kontextsensitive Grammatik  $G = (V, \Sigma, P, S)$  enthält dann neben den Start- und A-Regeln

$$S \rightarrow A(\hat{a}, a), A(\hat{b}, b), (q_0 \hat{a}, a), (q_0 \hat{b}, b) \quad (S_1-S_4)$$

$$A \rightarrow A(a, a), A(b, b), (q_0 a, a), (q_0 b, b) \quad (A_1-A_4)$$

für jedes Zeichen  $c \in \Gamma$  die F- und E-Regeln (wegen  $E = \{q_4\}$ )

$$(c, a) \rightarrow a, (c, b) \rightarrow b \quad (F_1-F_{16})$$

$$(q_4 c, a) \rightarrow a, (q_4 c, b) \rightarrow b \quad (E_1-E_{16})$$

# Beweis von $\{L(M) \mid M \text{ ist ein LBA}\} \subseteq \text{CSL}$

## Beispiel

- Die zugehörige kontextsensitive Grammatik  $G = (V, \Sigma, P, S)$  enthält dann neben den Start- und A-Regeln

$$S \rightarrow A(\hat{a}, a), A(\hat{b}, b), (q_0\hat{a}, a), (q_0\hat{b}, b) \quad (S_1-S_4)$$

$$A \rightarrow A(a, a), A(b, b), (q_0a, a), (q_0b, b) \quad (A_1-A_4)$$

für jedes Zeichen  $c \in \Gamma$  die F- und E-Regeln (wegen  $E = \{q_4\}$ )

$$(c, a) \rightarrow a, (c, b) \rightarrow b \quad (F_1-F_{16})$$

$$(q_4c, a) \rightarrow a, (q_4c, b) \rightarrow b \quad (E_1-E_{16})$$

- Daneben enthält  $P$  beispielsweise noch folgende Regeln:

- Für die Anweisung  $q_3\hat{B} \rightarrow q_4\hat{B}N$  die N-Regeln

$$(q_3\hat{B}, a) \rightarrow (q_4\hat{B}, a), (q_3\hat{B}, b) \rightarrow (q_4\hat{B}, b)$$

- Für die Anweisung  $q_1b \rightarrow q_2bL$  die L-Regeln (für jedes  $d \in \Gamma$ )

$$(d, a)(q_1b, a) \rightarrow (q_2d, a)(B, a), (d, b)(q_1b, a) \rightarrow (q_2d, b)(B, a)$$

$$(d, a)(q_1b, b) \rightarrow (q_2d, a)(B, b), (d, b)(q_1b, b) \rightarrow (q_2d, b)(B, b)$$

# Beweis von $\{L(M) \mid M \text{ ist ein LBA}\} \subseteq \text{CSL}$

## Beispiel

- Daneben enthält  $P$  beispielsweise noch folgende Regeln:

- Für die Anweisung  $q_3 \hat{B} \rightarrow q_4 \hat{B} N$  die N-Regeln

$$(q_3 \hat{B}, a) \rightarrow (q_4 \hat{B}, a), \quad (q_3 \hat{B}, b) \rightarrow (q_4 \hat{B}, b)$$

- Für die Anweisung  $q_1 b \rightarrow q_2 BL$  die L-Regeln (für jedes  $d \in \Gamma$ )

$$(d, a)(q_1 b, a) \rightarrow (q_2 d, a)(B, a), \quad (d, b)(q_1 b, a) \rightarrow (q_2 d, b)(B, a)$$
$$(d, a)(q_1 b, b) \rightarrow (q_2 d, a)(B, b), \quad (d, b)(q_1 b, b) \rightarrow (q_2 d, b)(B, b)$$

- Für die Anweisung  $q_0 a \rightarrow q_1 AR$  die R-Regeln (für jedes  $d \in \Gamma$ )

$$(q_0 a, a)(d, a) \rightarrow (A, a)(q_1 d, a), \quad (q_0 a, a)(d, b) \rightarrow (A, a)(q_1 d, b)$$
$$(q_0 a, b)(d, a) \rightarrow (A, b)(q_1 d, a), \quad (q_0 a, b)(d, b) \rightarrow (A, b)(q_1 d, b)$$

## Beweis von $\{L(M) \mid M \text{ ist ein LBA}\} \subseteq \text{CSL}$

- Sei  $M = (Z, \hat{\Sigma}, \Gamma, \delta, q_0, E)$  ein LBA (o.B.d.A. sei  $\varepsilon \notin L(M)$ ).
- Betrachte die kontextsensitive Grammatik  $G = (V, \Sigma, P, S)$  mit

$$V = \{S, A\} \cup (Z\Gamma \cup \Gamma) \times \Sigma,$$

die für alle  $a, b \in \Sigma$  und  $c, d \in \Gamma$  folgende Regeln enthält:

$S \rightarrow A(\hat{a}, a)$		(S)	„Startregeln“
$A \rightarrow A(a, a), (q_0 a, a)$		(A)	„A-Regeln“
$(c, a) \rightarrow a$		(F)	„Finale Regeln“
$(qc, a) \rightarrow a,$	falls $q \in E$	(E)	„E-Regeln“
$(qc, a) \rightarrow (q'c', a),$	falls $qc \rightarrow_M q'c'N$	(N)	„N-Regeln“
$(qc, a)(d, b) \rightarrow (c', a)(q'd, b),$	falls $qc \rightarrow_M q'c'R$	(R)	„R-Regeln“
$(d, a)(qc, b) \rightarrow (q'd, a)(c', b),$	falls $qc \rightarrow_M q'c'L$	(L)	„L-Regeln“

## Beweis von $\{L(M) \mid M \text{ ist ein LBA}\} \subseteq \text{CSL}$

- Durch Induktion über  $m$  lässt sich nun leicht für alle  $a_1, \dots, a_n \in \Gamma$  und  $q \in Z$  die folgende Äquivalenz beweisen:

$$q_0 x_1 \cdots x_{n-1} \hat{x}_n \vdash^m a_1 \cdots a_{i-1} q a_i \cdots a_n \text{ gdw.}$$

$$(q_0 x_1, x_1) \cdots (\hat{x}_n, x_n) \xRightarrow{(N,R,L)}^m (a_1, x_1) \cdots (q a_i, x_i) \cdots (a_n, x_n)$$

- Ist also  $q_0 x_1 \cdots x_{n-1} \hat{x}_n \vdash^m a_1 \cdots a_{i-1} q a_i \cdots a_n$  eine akzeptierende Rechnung von  $M(x_1 \cdots x_{n-1} \hat{x}_n)$  mit  $q \in E$ , so folgt

$$\begin{aligned} S &\xRightarrow{(S)} A(\hat{x}_n, x_n) \xRightarrow{(A)}^{n-1} (q_0 x_1, x_1)(x_2, x_2) \cdots (x_{n-1}, x_{n-1})(\hat{x}_n, x_n) \\ &\xRightarrow{(N,L,R)}^m (a_1, x_1) \cdots (a_{i-1}, x_{i-1})(q a_i, x_i) \cdots (a_n, x_n) \xRightarrow{(F,E)}^n x_1 \cdots x_n \end{aligned}$$

- Die Inklusion  $L(G) \subseteq L(M)$  folgt analog. □

### Bemerkung

Eine einfache Modifikation des Beweises zeigt, dass 1-NTMs genau die Sprachen vom Typ 0 akzeptieren (siehe Übungen).

# Deterministisch kontextsensitive Sprachen

## Definition

Die Klasse der **deterministisch kontextsensitiven** Sprachen ist definiert als

$$\text{DCSL} = \{L(M) \mid M \text{ ist ein DLBA}\}.$$

## Bemerkung

- Der DLBA  $M'$  für die Sprache  $\{a^n b^n \mid n \geq 1\}$  aus obigem Beispiel lässt sich leicht in einen DLBA für die kontextsensitive Sprache  $\{a^n b^n c^n \mid n \geq 1\}$  transformieren (siehe Übungen).
- Die Sprache  $\{a^n b^n c^n \mid n \geq 1\}$  liegt also in  $\text{DCSL} \setminus \text{CFL}$ .
- Bis heute ungelöst ist die Frage, ob die Klasse DCSL eine echte Teilklasse von CSL ist oder nicht?
- Diese Fragestellung ist als **LBA-Problem** bekannt.



## Zusammenfassung der Abschlusseigenschaften

	Vereinigung	Schnitt	Komplement	Produkt	Sternhülle
REG	ja	ja	ja	ja	ja
DCFL	nein	nein	ja	nein	nein
CFL	ja	nein	nein	ja	ja
DCSL	ja	ja	ja	ja	ja
CSL	ja	ja	ja	ja	ja
RE	ja	ja	nein	ja	ja

- In der VL Komplexitätstheorie wird gezeigt, dass die Klasse CSL unter Komplementbildung abgeschlossen ist.
- Im nächsten Kapitel werden wir sehen, dass die Klasse RE nicht unter Komplementbildung abgeschlossen ist.
- Die übrigen Abschlusseigenschaften der Klassen DCSL, CSL und RE in obiger Tabelle werden in den Übungen bewiesen.

# Entscheidbare und semi-entscheidbare Sprachen

## Definition

- Eine NTM  $M$  **hält** bei Eingabe  $x$ , falls alle Rechnungen von  $M(x)$  eine endliche Länge haben.
- Eine NTM  $M$  **entscheidet** eine Eingabe  $x$ , falls  $M(x)$  hält oder eine Konfiguration mit einem Endzustand erreicht.
- Eine Sprache  $L \subseteq \Sigma^*$  heißt **entscheidbar**, falls eine DTM  $M$  mit  $L(M) = L$  existiert, die jede Eingabe  $x \in \Sigma^*$  entscheidet.
- Jede von einer DTM  $M$  erkannte Sprache heißt **semi-entscheidbar**.

## Bemerkung

- Die von  $M$  akzeptierte Sprache  $L(M)$  heißt semi-entscheidbar, da  $M$  zwar alle Eingaben  $x \in L$  entscheidet (aber eventuell nicht alle  $x \in \bar{L}$ ).
- Später werden wir sehen, dass genau die Typ-0 Sprachen semi-entscheidbar sind.

# Berechnung von totalen Funktionen

## Definition

- Eine  $k$ -DTM  $M = (Z, \Sigma, \Gamma, \delta, q_0, E)$  **berechnet** eine Funktion  $f : \Sigma^* \rightarrow \Gamma^*$ , falls  $M$  bei jeder Eingabe  $x \in \Sigma^*$  in einer Konfiguration 
$$K = (q, u_1, a_1, v_1, \dots, u_k, a_k, v_k) \in Z \times (\Gamma^* \times \Gamma \times \Gamma^*)^k$$
 hält (d.h.  $K_x \vdash^* K$  und  $K$  hat keine Folgekonfiguration) mit 
$$u_k = f(x).$$
- Hierfür sagen wir auch,  $M$  gibt bei Eingabe  $x$  das Wort  $f(x)$  aus und schreiben  $M(x) = f(x)$ .
- $f$  heißt **Turing-berechenbar** (oder einfach **berechenbar**), falls es eine  $k$ -DTM  $M$  mit  $M(x) = f(x)$  für alle  $x \in \Sigma^*$  gibt.
- Aus historischen Gründen werden berechenbare Funktionen auch **rekursiv** genannt.

# Berechenbarkeit von partiellen Funktionen

## Definition

- Eine **partielle Funktion** hat die Form  $f : \Sigma^* \rightarrow \Gamma^* \cup \{\uparrow\}$ .
- Für  $f(x) = \uparrow$  sagen wir auch  $f(x)$  ist **undefiniert**.
- Der **Definitionsbereich** (engl. *domain*) von  $f$  ist

$$\text{dom}(f) = \{x \in \Sigma^* \mid f(x) \neq \uparrow\}.$$

- Das **Bild** (engl. *image*) von  $f$  ist

$$\text{img}(f) = \{f(x) \mid x \in \text{dom}(f)\}.$$

- Eine  $k$ -DTM  $M = (Z, \Sigma, \Gamma, \delta, q_0, E)$  **berechnet**  $f$ , falls  $M(x)$  für alle  $x \in \text{dom}(f)$  das Wort  $f(x)$  ausgibt und für alle  $x \notin \text{dom}(f)$  keine Ausgabe berechnet (d.h.  $M(x)$  darf im Fall  $x \notin \text{dom}(f)$  nicht halten).

# Berechen- und Entscheidbarkeit

Wir fassen die entscheidbaren Sprachen und die (partiellen) berechenbaren Funktionen in folgenden Klassen zusammen:

$REC = \{L(M) \mid M \text{ ist eine DTM, die jede Eingabe entscheidet}\},$

$FREC = \{f \mid f \text{ ist eine (totale) berechenbare Funktion}\},$

$FREC_p = \{f \mid f \text{ ist eine partielle berechenbare Funktion}\}.$

Dann gilt:

- $FREC \subsetneq FREC_p$  und
- $REG \subsetneq DCFL \subsetneq CFL \subsetneq DCSL \subseteq CSL \subsetneq REC \subsetneq RE.$

# Berechenbarkeit von partiellen Funktionen

## Beispiel

- Bezeichne  $x^+$  den **lexikografischen Nachfolger** von  $x \in \Sigma^*$ .
- Für  $\Sigma = \{0, 1\}$  ergeben sich beispielsweise folgende Werte:

$x$	$\varepsilon$	0	1	00	01	10	11	000	...
$x^+$	0	1	00	01	10	11	000	001	...

- Betrachte die auf  $\Sigma^*$  definierten Funktionen  $f_1, f_2, f_3, f_4$  mit

$$\begin{aligned} f_1(x) &= 0, \\ f_2(x) &= x, \\ f_3(x) &= x^+ \end{aligned} \quad \text{und} \quad f_4(x) = \begin{cases} \uparrow, & x = \varepsilon, \\ y, & x = y^+. \end{cases}$$

- Da  $f_1, f_2, f_3, f_4$  berechenbar sind, gehören die totalen Funktionen  $f_1, f_2, f_3$  zu  $\text{FREC}$  und die partielle Funktion  $f_4$  zu  $\text{FREC}_p$ .

# Berechenbarkeit von charakteristischen Funktionen

## Satz

- Eine Sprache  $A \subseteq \Sigma^*$  ist genau dann entscheidbar, wenn ihre **charakteristische Funktion**  $\chi_A : \Sigma^* \rightarrow \{0, 1\}$  berechenbar ist:

$$\chi_A(x) = \begin{cases} 1, & x \in A, \\ 0, & x \notin A. \end{cases}$$

- Eine Sprache  $A \subseteq \Sigma^*$  ist genau dann semi-entscheidbar, falls ihre **partielle charakteristische Funktion**  $\hat{\chi}_A : \Sigma^* \rightarrow \{0, 1, \uparrow\}$  berechenbar ist:

$$\hat{\chi}_A(x) = \begin{cases} 1, & x \in A, \\ \uparrow, & x \notin A. \end{cases}$$

## Beweis

Siehe Übungen.

# Charakterisierung der rekursiv aufzählbaren Sprachen

## Definition

Eine Sprache  $A \subseteq \Sigma^*$  heißt **rekursiv aufzählbar**, falls  $A$  leer oder das Bild  $\text{img}(f)$  einer berechenbaren Funktion  $f : \Gamma^* \rightarrow \Sigma^*$  ist.

## Satz

Folgende Eigenschaften sind äquivalent:

- 1  $A$  ist semi-entscheidbar (d.h.  $A$  wird von einer DTM akzeptiert),
- 2  $A$  wird von einer 1-DTM akzeptiert,
- 3  $A$  wird von einer 1-NTM akzeptiert,
- 4  $A$  ist vom Typ 0,
- 5  $A$  wird von einer NTM akzeptiert,
- 6  $A$  ist rekursiv aufzählbar.

## Beweis

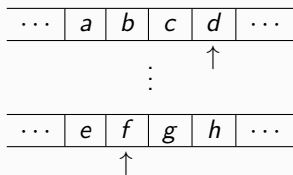
Die Implikation 2  $\Rightarrow$  3 ist klar. Die Implikationen 3  $\Rightarrow$  4  $\Rightarrow$  5 werden in den Übungen gezeigt. Hier zeigen wir 1  $\Rightarrow$  2 und 5  $\Rightarrow$  6  $\Rightarrow$  1.



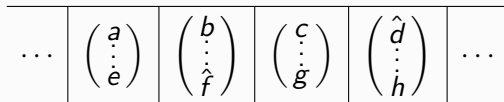
# Charakterisierung der rekursiv aufzählbaren Sprachen

Beweis von ①  $\Rightarrow$  ②:  $\{L(M) \mid M \text{ ist eine DTM}\} \subseteq \{L(M) \mid M \text{ ist eine 1-DTM}\}$

- Sei  $M = (Z, \Sigma, \Gamma, \delta, q_0, E)$  eine  $k$ -DTM mit  $L(M) = A$ .
- Wir konstruieren eine 1-DTM  $M' = (Z', \Sigma, \Gamma', \delta', z_0, E)$  für  $A$ .
- $M'$  simuliert  $M$ , indem sie jede Konfiguration  $K$  von  $M$  der Form



durch eine Konfiguration  $K'$  folgender Form nachbildet:



# Charakterisierung der rekursiv aufzählbaren Sprachen

Beweis von ①  $\Rightarrow$  ②:  $\{L(M) \mid M \text{ ist eine DTM}\} \subseteq \{L(M) \mid M \text{ ist eine 1-DTM}\}$

- Das heißt,  $M'$  arbeitet mit dem Alphabet

$$\Gamma' = \Gamma \cup (\Gamma \cup \{\hat{a} \mid a \in \Gamma\})^k$$

- und erzeugt bei Eingabe  $x = x_1 \cdots x_n \in \Sigma^*$  zuerst die der Startkonfiguration

$$K_x = (q_0, \varepsilon, x, \varepsilon, \sqcup, \dots, \varepsilon, \sqcup)$$

von  $M$  bei Eingabe  $x$  entsprechende Konfiguration

$$K'_x = q'_0 \begin{pmatrix} \hat{x}_1 \\ \hat{\sqcup} \\ \vdots \\ \hat{\sqcup} \end{pmatrix} \begin{pmatrix} x_2 \\ \sqcup \\ \vdots \\ \sqcup \end{pmatrix} \cdots \begin{pmatrix} x_n \\ \sqcup \\ \vdots \\ \sqcup \end{pmatrix}.$$

# Charakterisierung der rekursiv aufzählbaren Sprachen

Beweis von ①  $\Rightarrow$  ②:  $\{L(M) \mid M \text{ ist eine DTM}\} \subseteq \{L(M) \mid M \text{ ist eine 1-DTM}\}$

- Dann simuliert  $M'$  jeweils einen Schritt von  $M$  durch folgende Sequenz von Rechenschritten:
  - Zuerst geht  $M'$  solange nach rechts, bis sie alle mit  $\wedge$  markierten Zeichen (z.B.  $\hat{a}_1, \dots, \hat{a}_k$ ) gefunden hat.
  - Diese Zeichen speichert  $M'$  in ihrem Zustand.
  - Anschließend geht  $M'$  wieder nach links und realisiert dabei die durch  $\delta(q, a_1, \dots, a_k)$  vorgegebene Anweisung von  $M$ .
  - Den aktuellen Zustand  $q$  von  $M$  speichert  $M'$  ebenfalls in ihrem Zustand.
- Sobald  $M$  in einen Endzustand übergeht, wechselt  $M'$  ebenfalls in einen Endzustand und hält.
- Nun ist leicht zu sehen, dass  $L(M') = L(M)$  ist. □

# Charakterisierung der rekursiv aufzählbaren Sprachen

Beweis von ⑤  $\Rightarrow$  ⑥:  $\{L(M) \mid M \text{ ist eine NTM}\} \subseteq \{L \mid L \text{ ist rek. aufzählbar}\}$

- Sei  $M = (Z, \Sigma, \Gamma, \delta, q_0, E)$  eine  $k$ -NTM und sei  $A = L(M) \neq \emptyset$ .
- Sei  $\tilde{\Gamma}$  das Alphabet  $Z \cup \Gamma \cup \{\#\}$ .
- Wir kodieren eine Konfiguration  $K = (q, u_1, a_1, v_1, \dots, u_k, a_k, v_k)$  durch das Wort

$$\text{code}(K) = \#q\#u_1\#a_1\#v_1\#\dots\#u_k\#a_k\#v_k\#$$

und eine Rechnung  $K_0 \vdash \dots \vdash K_t$  durch  $\text{code}(K_0) \dots \text{code}(K_t)$ .

- Dann lassen sich die Wörter von  $A$  durch folgende Funktion  $f : \tilde{\Gamma}^* \rightarrow \Sigma^*$  aufzählen (dabei ist  $x_0$  ein beliebiges Wort in  $A$ ):

$$f(x) = \begin{cases} y, & x \text{ kodiert eine Rechnung } K_0 \vdash \dots \vdash K_t \text{ von } M \\ & \text{mit } K_0 = K_y \text{ und } K_t \in E \times (\Gamma^* \times \Gamma \times \Gamma^*)^k \\ x_0, & \text{sonst.} \end{cases}$$

- Da  $f$  berechenbar ist, ist  $A = \text{img}(f)$  rekursiv aufzählbar. □

# Charakterisierung der rekursiv aufzählbaren Sprachen

Beweis von ⑥  $\Rightarrow$  ①:  $\{L \mid L \text{ ist rek. aufzählbar}\} \subseteq \{L(M) \mid M \text{ ist eine DTM}\}$

- Sei  $f : \Gamma^* \rightarrow \Sigma^*$  eine Funktion mit  $A = \text{img}(f)$  und sei  $M$  eine  $k$ -DTM, die  $f$  berechnet.
- Betrachte folgende  $(k + 1)$ -DTM  $M'$ , die bei Eingabe  $x$ 
  - auf dem 2. Band der Reihe nach alle Wörter  $y$  in  $\Gamma^*$  erzeugt,
  - für jedes  $y$  den Wert  $f(y)$  durch Simulation von  $M(y)$  berechnet, und
  - ihre Eingabe  $x$  akzeptiert, sobald  $f(y) = x$  ist. □

# Charakterisierung der entscheidbaren Sprachen

## Satz

$A$  ist genau dann entscheidbar, wenn  $A$  und  $\bar{A}$  semi-entscheidbar sind, d.h.  
 $REC = RE \cap \text{co-RE}$ .

## Beweis.

- Falls  $A$  entscheidbar ist, ist auch  $\bar{A}$  entscheidbar, d.h.  $A$  und  $\bar{A}$  sind dann auch semi-entscheidbar.
- Für die Rückrichtung seien  $f_1, f_2 : \Gamma^* \rightarrow \Sigma^*$  Turing-berechenbare Funktionen mit  $\text{img}(f_1) = A$  und  $\text{img}(f_2) = \bar{A}$ .
- Wir betrachten folgende DTM  $M$ , die bei Eingabe  $x$ 
  - für jedes  $y \in \Gamma^*$  die beiden Werte  $f_1(y)$  und  $f_2(y)$  bestimmt,
  - $x$  akzeptiert, sobald ein  $y$  auf den Wert  $f_1(y) = x$  führt und
  - $x$  verwirft, sobald ein  $y$  auf den Wert  $f_2(y) = x$  führt.
- Da jede Eingabe  $x$  entweder in  $\text{img}(f_1) = A$  oder in  $\text{img}(f_2) = \bar{A}$  enthalten ist, entscheidet  $M$  alle Eingaben. □

# Kodierung (Gödelisierung) von Turingmaschinen

- Sei  $M = (Z, \Sigma, \Gamma, \delta, q_0, E)$  eine 1-DTM mit
  - Zustandsmenge  $Z = \{q_0, \dots, q_m\}$  (o.B.d.A. sei  $E = \{q_m\}$ ),
  - Eingabealphabet  $\Sigma = \{0, 1, \#\}$  und
  - Arbeitsalphabet  $\Gamma = \{a_0, \dots, a_l\}$ , wobei wir o.B.d.A.  $a_0 = 0$ ,  $a_1 = 1$ ,  $a_2 = \#$ ,  $a_3 = \sqcup$  annehmen.
- Dann können wir jede Anweisung der Form  $q_i a_j \rightarrow q_{i'} a_{j'} D$  durch das Wort

$$\# \text{bin}(i) \# \text{bin}(j) \# \text{bin}(i') \# \text{bin}(j') \# b_D \#$$

kodieren.

- Dabei ist  $\text{bin}(n)$  die Binärdarstellung von  $n$  und

$$b_D = \begin{cases} 0, & D = N, \\ 1, & D = L, \\ 10, & D = R. \end{cases}$$

# Kodierung von Turingmaschinen

- $M$  lässt sich nun als ein Wort über dem Alphabet  $\{0, 1, \#\}$  kodieren, indem wir die Anweisungen von  $M$  in kodierter Form auflisten.
- Kodieren wir die Zeichen  $0, 1, \#$  binär (z.B.  $0 \mapsto 00, 1 \mapsto 11, \# \mapsto 10$ ), so gelangen wir zu einer Binärkodierung  $w_M$  von  $M$ .
- Die Binärzahl  $w_M$  wird auch die **Gödel-Nummer** von  $M$  genannt.
- $M_w$  ist durch Angabe von  $w$  bis auf die Benennung ihrer Zustände und Arbeitszeichen eindeutig bestimmt.
- Ganz analog lassen sich auch DTMs mit einer beliebigen Anzahl von Bändern (sowie NTMs, Konfigurationen oder Rechnungen von TMs) kodieren.
- Umgekehrt können wir jedem Binärstring  $w \in \{0, 1\}^*$  eine DTM  $M_w$  wie folgt zuordnen:

$$M_w = \begin{cases} M, & \text{falls eine DTM } M \text{ mit } w_M = w \text{ existiert,} \\ M_0, & \text{sonst (dabei sei } M_0 \text{ eine beliebige DTM).} \end{cases}$$



# Unentscheidbarkeit des Halteproblems

## Definition

- Das **Halteproblem** ist die Sprache

$$H = \left\{ w \# x \mid \begin{array}{l} w, x \in \{0, 1\}^* \text{ und} \\ \text{die DTM } M_w \text{ h\u00e4lt} \\ \text{bei Eingabe } x \end{array} \right\}$$

- Das **spezielle Halteproblem** ist

$$K = \left\{ w \in \{0, 1\}^* \mid \begin{array}{l} \text{die DTM } M_w \\ \text{h\u00e4lt bei Eingabe } w \end{array} \right\}$$

$\chi_H$	$x_1$	$x_2$	$x_3$	$\dots$
$w_1$	0	1	0	$\dots$
$w_2$	0	1	1	$\dots$
$w_3$	1	1	0	$\dots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

$\chi_K$				
$w_1$	0			
$w_2$		1		
$w_3$			0	
$\vdots$				$\ddots$

## Satz

$K \in \text{RE} \setminus \text{REC}$ .

## Semi-Entscheidbarkeit des speziellen Halteproblems

### Beweis von $K \in RE$

- Sei  $w_0$  die Kodierung einer DTM, die bei jeder Eingabe (sofort) hält und betrachte die Funktion

$$f(x) = \begin{cases} w, & x \text{ ist Kodierung einer haltenden Rechnung einer} \\ & \text{DTM } M_w \text{ bei Eingabe } w, \\ w_0, & \text{sonst.} \end{cases}$$

- Da  $f$  berechenbar und  $\text{img}(f) = K$  ist, folgt  $K \in RE$ . □

### Bemerkung

Ganz ähnlich lässt sich  $H \in RE$  zeigen.

# Unentscheidbarkeit des speziellen Halteproblems

## Beweis von $K \notin \text{REC}$

- Angenommen, die Sprache

$$K = \{w \mid M_w(w) \text{ h\"alt}\} \quad (*)$$

w\"are durch eine DTM  $M_K$  entscheidbar.

- Betrachte die DTM  $\hat{M}$ , die bei Eingabe  $w \in \{0,1\}^*$  die DTM  $M_K(w)$  simuliert und genau dann h\"alt, wenn  $M_K(w)$  verwirft:

$$\hat{M}(w) \text{ h\"alt} \Leftrightarrow w \notin K \quad (**)$$

- F\"ur die Kodierung  $\hat{w}$  von  $\hat{M}$  folgt dann aber

$$\hat{w} \in K \stackrel{(*)}{\Leftrightarrow} M_{\hat{w}}(\hat{w}) \text{ h\"alt} \stackrel{(**)}{\Leftrightarrow} \hat{w} \notin K \quad \text{! (Widerspruch!)}$$

$\chi_H$	$x_1$	$x_2$	$x_3$	$\dots$
$w_1$	0	1	0	$\dots$
$w_2$	0	1	1	$\dots$
$w_3$	1	1	0	$\dots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

$\hat{w}$	1	0	1	$\dots$
-----------	---	---	---	---------

# Unentscheidbarkeit des speziellen Halteproblems

## Korollar

- $\text{REC} \subsetneq \text{RE}$ ,
- $K \in \text{RE} \setminus \text{co-RE}$  (d.h.  $\text{RE} \neq \text{co-RE}$ ),
- $\bar{K} \in \text{co-RE} \setminus \text{RE}$ .

## Beweis

- $\text{REC} \subsetneq \text{RE}$ : klar, da  $K \in \text{RE} - \text{REC}$ .
- $K \notin \text{co-RE}$ : Aus der Annahme  $K \in \text{co-RE}$  würde wegen  $K \in \text{RE}$  folgen, dass  $K$  entscheidbar ist (Widerspruch).
- $\bar{K} \in \text{co-RE} \setminus \text{RE}$ : klar, da  $K \in \text{RE} \setminus \text{co-RE}$ . □

# Der Reduktionsbegriff

## Definition

Eine Sprache  $A \subseteq \Sigma^*$  heißt auf  $B \subseteq \Gamma^*$  **reduzierbar** (kurz:  $A \leq B$ ), falls eine berechenbare Funktion  $f : \Sigma^* \rightarrow \Gamma^*$  ex., so dass gilt:

$$\forall x \in \Sigma^* : x \in A \Leftrightarrow f(x) \in B.$$

## Beispiel

Es gilt  $K \leq H$  mittels  $f : w \mapsto w\#w$ , da für alle  $w \in \{0,1\}^*$  gilt:

$$\begin{aligned} w \in K &\Leftrightarrow M_w \text{ ist eine DTM, die bei Eingabe } w \text{ hält} \\ &\Leftrightarrow w\#w \in H. \end{aligned}$$



# Abschluss von REC unter $\leq$

## Definition

- Eine Sprachklasse  $\mathcal{C}$  heißt **unter  $\leq$  abgeschlossen**, wenn für alle Sprachen  $A, B$  gilt:

$$A \leq B \wedge B \in \mathcal{C} \Rightarrow A \in \mathcal{C}.$$

## Satz

Die Klassen REC und RE sind unter  $\leq$  abgeschlossen.

## Beweis

- Gelte  $A \leq B$  mittels  $f$  und sei  $B \in \text{REC}$ .
- Dann ex. eine DTM  $M$ , die  $\chi_B$  berechnet.
- Betrachte folgende DTM  $M'$ :
  - $M'$  berechnet bei Eingabe  $x$  zuerst den Wert  $f(x)$  und
  - simuliert dann  $M$  bei Eingabe  $f(x)$ .

## Abschluss von REC und RE unter $\leq$

### Satz

Die Klasse REC ist unter  $\leq$  abgeschlossen.

### Beweis.

- Gelte  $A \leq B$  mittels  $f$  und sei  $B \in \text{REC}$ .
- Dann ex. eine DTM  $M$ , die  $\chi_B$  berechnet.
- Betrachte folgende DTM  $M'$ :
  - $M'$  berechnet bei Eingabe  $x$  zuerst den Wert  $f(x)$  und
  - simuliert dann  $M$  bei Eingabe  $f(x)$ .
- Wegen  $x \in A \Leftrightarrow f(x) \in B$  ist  $\chi_A(x) = \chi_B(f(x))$  und daher folgt
$$M'(x) = M(f(x)) = \chi_B(f(x)) = \chi_A(x).$$
- Also berechnet  $M'$  die Funktion  $\chi_A$ , d.h.  $A \in \text{REC}$ . □

### Bemerkung

Der Abschluss von RE unter  $\leq$  folgt analog (siehe Übungen).

# Der Vollständigkeitsbegriff

## Definition

- Eine Sprache  $A$  heißt **hart** für eine Sprachklasse  $\mathcal{C}$  (kurz:  **$\mathcal{C}$ -hart** oder  **$\mathcal{C}$ -schwer**), falls jede Sprache  $L \in \mathcal{C}$  auf  $A$  reduzierbar ist:

$$\forall L \in \mathcal{C} : L \leq A.$$

- Eine  $\mathcal{C}$ -harte Sprache  $A$ , die zu  $\mathcal{C}$  gehört, heißt  **$\mathcal{C}$ -vollständig**.

## Beispiel

Das Halteproblem  $H$  ist RE-vollständig. Es gilt nämlich

- $H \in \text{RE}$  und
- $\forall L \in \text{RE} : L \leq H$

mittels der Reduktionsfunktion  $x \mapsto w\#x$ , wobei  $w$  die Kodierung einer DTM  $M_w$  ist, die  $\hat{\chi}_L$  berechnet.



## Bemerkung

Auch das spezielle Halteproblem  $K$  ist RE-vollständig (siehe Übungen).



# $H$ ist nicht entscheidbar

## Korollar

- $A \leq B \wedge A \notin \text{REC} \Rightarrow B \notin \text{REC}.$
- $A \leq B \wedge A \notin \text{RE} \Rightarrow B \notin \text{RE}.$

## Beweis

Aus der Annahme, dass  $B$  entscheidbar (bzw. semi-entscheidbar) ist, folgt wegen  $A \leq B$ , dass dies auch auf  $A$  zutrifft (Widerspruch).  $\square$

## Bemerkung

Wegen  $K \leq H$  überträgt sich somit die Unentscheidbarkeit von  $K$  auf  $H$ .

## Korollar

$H \notin \text{REC}.$

# Das Halteproblem bei leerem Band

## Definition

Das **Halteproblem bei leerem Band** ist die Sprache

$$H_0 = \left\{ w \in \{0, 1\}^* \mid \begin{array}{l} \text{die DTM } M_w \\ \text{hält bei Eingabe } \varepsilon \end{array} \right\}$$

## Satz

$H_0$  ist RE-vollständig.

## Beweis

- $H_0 \in \text{RE}$  folgt wegen  $H_0 \leq H \in \text{RE}$  mittels der Reduktionsfunktion  $w \mapsto w\#\varepsilon$ .

$\chi_H$	$x_1$	$x_2$	$x_3$	$\dots$
$w_1$	0	1	0	$\dots$
$w_2$	0	1	1	$\dots$
$w_3$	1	1	0	$\dots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

$\chi_{H_0}$	$x_1 (= \varepsilon)$
$w_1$	0
$w_2$	0
$w_3$	1
$\vdots$	$\vdots$

## $H$ ist auf $H_0$ reduzierbar

### Beweis

- $H_0 \in \text{RE}$  folgt wegen  $H_0 \leq H \in \text{RE}$  mittels der Reduktionsfunktion  $w \mapsto w\#\varepsilon$ .
- Sei  $A \in \text{RE}$  und sei  $w$  die Kodierung einer DTM, die  $\hat{\chi}_A$  berechnet. Um  $A$  auf  $H_0$  zu reduzieren, transformieren wir  $x \in \{0, 1\}^*$  auf die Kodierung einer DTM  $M_{w_x}$ , die zunächst ihre Eingabe durch  $x$  ersetzt und dann  $M_w(x)$  simuliert. Dann gilt

$$x \in A \iff w_x \in H_0$$

und somit  $A \leq H_0$  mittels der Reduktionsfunktion  $x \mapsto w_x$ .

### Korollar

$H_0 \notin \text{REC}$ .

# Der Satz von Rice

## Frage

- Kann man einer beliebig vorgegebenen DTM ansehen, ob die von ihr berechnete Funktion eine gewisse Eigenschaft hat?
- Kann man beispielsweise entscheiden, ob eine gegebene DTM eine totale Funktion berechnet?

## Antwort

Nein (es sei denn, die fragliche Eigenschaft ist trivial, d.h. keine oder jede DTM berechnet eine Funktion mit dieser Eigenschaft).

# Der Satz von Rice

## Definition

- Eine Eigenschaft  $\mathcal{F}$  von Funktionen heißt **trivial**, wenn  $\mathcal{F}$  entweder alle oder keine partielle berechenbare Funktionen auf  $\{0, 1, \#\}^*$  enthält.
- Zu  $\mathcal{F}$  definieren wir die Sprache

$$L_{\mathcal{F}} = \left\{ w \in \{0, 1\}^* \mid \text{die DTM } M_w \text{ berechnet eine Funktion in } \mathcal{F} \right\}.$$

$\mathcal{F}$  ist also nicht trivial, wenn  $L_{\mathcal{F}} \neq \emptyset$  und  $L_{\mathcal{F}} \neq \{0, 1\}^*$  ist.

Der Satz von Rice besagt, dass  $L_{\mathcal{F}}$  in diesem Fall unentscheidbar ist.

## Satz (Satz von Rice)

Für jede nicht triviale Eigenschaft  $\mathcal{F}$  ist  $L_{\mathcal{F}}$  unentscheidbar.

# Der Satz von Rice

## Beispiel

- Die Sprache

$$L = \{w \in \{0, 1\}^* \mid M_w(0^n) = 0^{n+1} \text{ für alle } n \geq 0\}$$

ist unentscheidbar.

- Dies folgt aus dem Satz von Rice, da  $L = L_{\mathcal{F}}$  für folgende nichttriviale Eigenschaft ist:

$$\mathcal{F} = \{f \in \text{FREC}_p \mid f(0^n) = 0^{n+1} \text{ für alle } n \geq 0\}.$$

- Es gibt nämlich partielle berechenbare Funktionen auf  $\{0, 1, \#\}^*$  mit dieser Eigenschaft wie z.B. die Funktion

$$f(x) = \begin{cases} 0^{n+1}, & x = 0^n \\ \uparrow, & \text{sonst} \end{cases}$$

als auch ohne diese Eigenschaft (wie die konstante Funktion  $g(x) = 0$ ).

# Der Satz von Rice

## Satz (Satz von Rice)

Für jede nicht triviale Eigenschaft  $\mathcal{F}$  ist die Sprache  $L_{\mathcal{F}}$  unentscheidbar.

## Beweis

- Die Idee besteht darin,  $H_0$  auf  $L_{\mathcal{F}}$  zu reduzieren, indem wir für eine gegebene DTM  $M_w$  eine DTM  $M_{w'}$  konstruieren mit
$$w \in H_0 \Leftrightarrow M_{w'} \text{ berechnet eine Funktion in } \mathcal{F}.$$
- Hierzu lassen wir  $M_{w'}$  bei Eingabe  $x$  zunächst einmal die DTM  $M_w$  bei Eingabe  $\varepsilon$  simulieren.
- Falls  $w \notin H_0$  ist, berechnet  $M_{w'}$  also die überall undefinierte Funktion  $u$  mit  $u(x) = \uparrow$  für alle  $x \in \{0, 1, \#\}^*$ .
- Für das Folgende nehmen wir an, dass  $u \notin \mathcal{F}$  ist. Andernfalls zeigen wir für die komplementäre Eigenschaft  $\mathcal{F}' = \overline{\mathcal{F}}$ , dass  $L_{\mathcal{F}'}$  unentscheidbar ist. Dies impliziert, dass auch  $\overline{L_{\mathcal{F}'}} = \overline{\overline{L_{\mathcal{F}}}} = L_{\mathcal{F}}$  unentscheidbar ist.
- Damit die Reduktion gelingt, müssen wir also nur dafür sorgen, dass  $M_{w'}$  im Fall  $w \in H_0$  eine Funktion  $f \in \mathcal{F}$  berechnet.

# Der Satz von Rice

## Beweis (Schluss)

- Damit die Reduktion gelingt, müssen wir also nur dafür sorgen, dass  $M_{w'}$  im Fall  $w \in H_0$  eine Funktion  $f \in \mathcal{F}$  berechnet.
- Da  $\mathcal{F}$  nicht trivial ist, gibt es eine DTM  $M$ , die eine partielle Funktion  $f \in \mathcal{F}$  berechnet.
- Betrachte die Reduktionsfunktion
$$h(w) = w', \text{ wobei } w' \text{ die Kodierung einer DTM ist, die bei Eingabe } x \text{ zunächst die DTM } M_w(\varepsilon) \text{ simuliert und im Fall, dass } M_w(\varepsilon) \text{ hält, mit der Simulation von } M(x) \text{ fortfährt.}$$
- Dann ist  $h : w \mapsto w'$  eine totale berechenbare Funktion und es gilt
$$\begin{aligned} w \in H_0 &\Rightarrow M_{w'} \text{ berechnet } f &\Rightarrow w' \in L_{\mathcal{F}}, \\ w \notin H_0 &\Rightarrow M_{w'} \text{ berechnet } u &\Rightarrow w' \notin L_{\mathcal{F}}. \end{aligned}$$
- Dies zeigt, dass  $h$  das Problem  $H_0$  auf  $L_{\mathcal{F}}$  reduziert, und da  $H_0$  unentscheidbar ist, muss auch  $L_{\mathcal{F}}$  unentscheidbar sein. □



# Berechen- und Entscheidbarkeit

Wir fassen die (semi-) entscheidbaren Sprachen und die (partiellen) berechenbaren Funktionen in folgenden Klassen zusammen:

$$\text{RE} = \{L(M) \mid M \text{ ist eine DTM}\},$$

$$\text{REC} = \{L(M) \mid M \text{ ist eine DTM, die jede Eingabe entscheidet}\},$$

$$\text{FREC} = \{f \mid f \text{ ist eine (totale) berechenbare Funktion}\},$$

$$\text{FREC}_p = \{f \mid f \text{ ist eine partielle berechenbare Funktion}\}.$$

Dann gilt:

- $\text{FREC} \subsetneq \text{FREC}_p$  und
- $\text{REG} \subsetneq \text{DCFL} \subsetneq \text{CFL} \subsetneq \text{DCSL} \subseteq \text{CSL} \subsetneq \text{REC} \subsetneq \text{RE}.$

# Das Postsche Korrespondenzproblem (PCP)

## Definition

- Sei  $\Sigma$  ein beliebiges Alphabet mit  $\# \notin \Sigma$ .
- Das **Postsche Korrespondenzproblem über  $\Sigma$**  (kurz **PCP $_{\Sigma}$** ) ist:  
gegeben:  $k$  Paare  $(x_1, y_1), \dots, (x_k, y_k)$  von Wörtern über  $\Sigma$ .  
gefragt: Gibt es eine Folge  $\alpha = (i_1, \dots, i_n)$ ,  $n \geq 1$ , von Indizes  $i_j \in \{1, \dots, k\}$  mit  $x_{i_1} \cdots x_{i_n} = y_{i_1} \cdots y_{i_n}$ ?
- Das **modifizierte PCP über  $\Sigma$**  (kurz **MPCP $_{\Sigma}$** ) fragt nach einer Lösung  $\alpha = (i_1, \dots, i_n)$  mit  $i_1 = 1$ .
- Wir notieren eine PCP-Instanz meist in Form einer Matrix  $\begin{pmatrix} x_1 \cdots x_k \\ y_1 \cdots y_k \end{pmatrix}$  und kodieren sie durch das Wort  $x_1 \# y_1 \# \cdots \# x_k \# y_k$ .

## Beispiel

Die Instanz  $I = \begin{pmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{pmatrix} = \begin{pmatrix} a & ab & caa \\ aca & bc & aa \end{pmatrix}$  besitzt wegen

$$x_1 x_3 x_2 x_3 = acaabcaa$$

$$y_1 y_3 y_2 y_3 = acaabcaa$$

die PCP-Lösung  $\alpha = (1, 3, 2, 3)$ , die auch eine MPCP-Lösung ist.

# Das Postsche Korrespondenzproblem

## Lemma

Für jedes Alphabet  $\Sigma$  gilt  $\text{PCP}_\Sigma \leq \text{PCP}_{\{a,b\}}$ .

## Beweis

- Sei  $\Sigma = \{a_1, \dots, a_m\}$ . Für ein Zeichen  $a_i \in \Sigma$  sei  $\hat{a}_i = ab^{i-1}$  und für ein Wort  $w = w_1 \cdots w_n \in \Sigma^*$  mit  $w_i \in \Sigma$  sei  $\hat{w} = \hat{w}_1 \cdots \hat{w}_n$ .
- Dann folgt  $\text{PCP}_\Sigma \leq \text{PCP}_{\{a,b\}}$  mittels der Reduktionsfunktion

$$f : \begin{pmatrix} x_1 \cdots x_k \\ y_1 \cdots y_k \end{pmatrix} \mapsto \begin{pmatrix} \hat{x}_1 \cdots \hat{x}_k \\ \hat{y}_1 \cdots \hat{y}_k \end{pmatrix}.$$

□

## Beispiel

Sei  $\Sigma = \{0, 1, 2\}$ . Dann ist  $\hat{0} = a$ ,  $\hat{1} = ab$  und  $\hat{2} = abb$ . Somit ist

$$f \left( \begin{pmatrix} 0 & 01 & 200 \\ 020 & 12 & 00 \end{pmatrix} \right) = \begin{pmatrix} a & aab & abbba \\ aabba & ababb & aa \end{pmatrix}.$$

# Das Postsche Korrespondenzproblem

Im Folgenden lassen wir im Fall  $\Sigma = \{a, b\}$  den Index weg und schreiben einfach PCP (bzw. MPCP).

## Satz

$\text{MPCP} \leq \text{PCP}$ .

## Beweis

- Wir zeigen  $\text{MPCP} \leq \text{PCP}_\Sigma$  für  $\Sigma = \{a, b, \langle, |, \rangle\}$ .
- Für ein Wort  $w = w_1 \cdots w_n$  sei

$$\begin{array}{cccc} \overleftarrow{w} & \overleftarrow{w} & \overleftarrow{w} & \overleftarrow{w} \\ \langle w_1 | \cdots | w_n \rangle & \langle w_1 | \cdots | w_n \rangle & | w_1 | \cdots | w_n & w_1 | \cdots | w_n \rangle \end{array}$$

# Das Postsche Korrespondenzproblem

## Beweis von MPCP $\leq$ PCP

- Wir zeigen MPCP  $\leq$  PCP $_{\Sigma}$  für  $\Sigma = \{a, b, \langle, |, \rangle\}$ .
- Für ein Wort  $w = w_1 \cdots w_n$  sei

$$\begin{array}{cccc} \overleftarrow{w} & \overleftarrow{w} & \overleftarrow{w} & \overleftarrow{w} \\ \hline \langle w_1 | \cdots | w_n \rangle & w_1 | \cdots | w_n & \langle w_1 | \cdots | w_n & | w_1 | \cdots | w_n \end{array}$$

- Wir reduzieren MPCP mittels folgender Funktion  $f$  auf PCP $_{\Sigma}$ :

$$f : \begin{pmatrix} x_1 & \cdots & x_k \\ y_1 & \cdots & y_k \end{pmatrix} \mapsto \begin{pmatrix} \overleftarrow{x_1} & \overleftarrow{x_1} & \cdots & \overleftarrow{x_k} & \rangle \\ \overleftarrow{y_1} & \overleftarrow{y_1} & \cdots & \overleftarrow{y_k} & | \end{pmatrix}$$

## Beispiel

$$f : \begin{pmatrix} aa & b & bab & bb \\ aab & bb & a & b \end{pmatrix} \mapsto \begin{pmatrix} \langle a|a & a|a & b| & b|a|b| & b|b| & \rangle \\ \langle a|a|b & |a|a|b & |b|b & |a & |b & | \end{pmatrix}$$

# Das Postsche Korrespondenzproblem

## Beweis von $\text{MPCP} \leq \text{PCP}$

- Wir zeigen  $\text{MPCP} \leq \text{PCP}_\Sigma$  für  $\Sigma = \{a, b, \langle, |, \rangle\}$ .
- Für ein Wort  $w = w_1 \cdots w_n$  sei

$$\begin{array}{cccc} \overleftarrow{w} & \overleftarrow{w} & \overleftarrow{w} & \overleftarrow{w} \\ \hline \langle w_1 | \cdots | w_n \rangle & w_1 | \cdots | w_n & \langle w_1 | \cdots | w_n & | w_1 | \cdots | w_n \end{array}$$

- Wir reduzieren MPCP mittels folgender Funktion  $f$  auf  $\text{PCP}_\Sigma$ :

$$f : \begin{pmatrix} x_1 & \cdots & x_k \\ y_1 & \cdots & y_k \end{pmatrix} \mapsto \begin{pmatrix} \overleftarrow{x_1} & \overleftarrow{x_1} & \cdots & \overleftarrow{x_k} & \rangle \\ \overleftarrow{y_1} & \overleftarrow{y_1} & \cdots & \overleftarrow{y_k} & | \rangle \end{pmatrix}$$

- Da jede MPCP-Lösung  $\alpha = (1, i_2, \dots, i_n)$  für  $I$  auf eine PCP-Lösung  $\alpha' = (1, i_2 + 1, \dots, i_n + 1, k + 2)$  für  $f(I)$  führt, folgt

$$I \in \text{MPCP} \Rightarrow f(I) \in \text{PCP}_\Sigma.$$

# Das Postsche Korrespondenzproblem

## Beweis von $\text{MPCP} \leq \text{PCP}$

- Für die umgekehrte Implikation sei  $\alpha' = (i_1, \dots, i_n)$  eine PCP-Lösung für

$$f(I) = \left( \begin{array}{cccc} \overleftarrow{x_1} & \overleftarrow{x_1} & \cdots & \overleftarrow{x_k} & \rangle \\ \overleftarrow{y_1} & \overleftarrow{y_1} & \cdots & \overleftarrow{y_k} & | \rangle \end{array} \right).$$

- Dann muss  $i_1 = 1$  sein, da  $(\overleftarrow{x_1}, \overleftarrow{y_1})$  das einzige Paar in  $f(I)$  ist, bei dem beide Komponenten mit demselben Buchstaben anfangen.
- Zudem muss  $i_n = k + 2$  sein, da nur das Paar  $(\rangle, | \rangle)$  mit demselben Buchstaben aufhört.
- Wählen wir  $\alpha'$  von minimaler Länge, so ist  $i_j \in \{2, \dots, k + 1\}$  für  $j = 2, \dots, n - 1$ .
- Dann ist aber

$$\alpha = (i_1, i_2 - 1, \dots, i_{n-1} - 1)$$

eine MPCP-Lösung für  $I$ .



# Das Postsche Korrespondenzproblem

## Satz

PCP ist RE-vollständig und damit unentscheidbar.

## Beweis.

- Es ist leicht zu sehen, dass  $PCP \in RE$  ist.
- Sei  $A \in RE$  und sei  $M = (Z, \Sigma, \Gamma, \delta, z_0, E)$  eine 1-DTM für  $A$ .
- Wir zeigen  $A \leq MPCP_{\Sigma'}$  für  $\Sigma' = \Gamma \cup Z \cup \{\langle, |, \rangle\}$ .
- Wegen  $MPCP_{\Sigma'} \leq PCP$  folgt hieraus  $A \leq PCP$ .

## Beweisidee für die Reduktion $A \leq MPCP_{\Sigma'}$ :

Transformiere eine Eingabe  $w \in \Sigma^*$  in eine Instanz  $f(w) = \begin{pmatrix} x_1 \cdots x_k \\ y_1 \cdots y_k \end{pmatrix}$ , so dass  $\alpha = (i_1, \dots, i_n)$  genau dann eine MPCP-Lösung für  $f(w)$  ist, wenn das zugehörige Lösungswort  $x_{i_1} \cdots x_{i_n} = y_{i_1} \cdots y_{i_n}$  eine akzeptierende Rechnung von  $M(w)$  kodiert. Dann gilt

$$x \in A \Leftrightarrow f(w) \in MPCP_{\Sigma'}.$$



# Das Postsche Korrespondenzproblem

## Beweis von $A \leq \text{MPCP}_{\Sigma'}$

Wir bilden  $f(w)$  aus folgenden Wortpaaren:

- 1  $(\langle, \langle | z_0 w)$ , „Startregel“
- 2 für alle  $a \in \Gamma \cup \{|\}$ :  $(a, a)$ , „Kopierregeln“
- 3 für alle  $a, a', b \in \Gamma, z, z' \in Z$ :
  - $(za, z'a')$ , falls  $\delta(z, a) = (z', a', N)$ ,
  - $(za, a'z')$ , falls  $\delta(z, a) = (z', a', R)$ ,
  - $(bza, z'ba')$ , falls  $\delta(z, a) = (z', a', L)$ ,
  - $(|za, |z' \sqcup a')$ , falls  $\delta(z, a) = (z', a', L)$ ,
  - $(bz|, z'ba'|)$ , falls  $\delta(z, \sqcup) = (z', a', L)$ ,
  - $(z|, z'a'|)$ , falls  $\delta(z, \sqcup) = (z', a', N)$ ,
  - $(z|, a'z'|)$ , falls  $\delta(z, \sqcup) = (z', a', R)$ ,
- 4 für alle  $e \in E, a \in \Gamma$ :  $(ae, e), (ea, e)$ , „Löschregeln“
- 5 sowie für alle  $e \in E$ :  $(e|, |e)$ . „Abschlussregeln“

# Das Postsche Korrespondenzproblem

## Beispiel

- Betrachte die 1-DTM  $M = (Z, \Sigma, \Gamma, \delta, q_0, E)$  mit  $Z = \{q_0, \dots, q_4\}$ ,  $\Sigma = \{a, b\}$ ,  $\Gamma = \{a, b, A, B, \sqcup\}$ ,  $E = \{q_4\}$  und den Anweisungen

$$\begin{aligned} \delta: & q_0a \rightarrow q_1AR \quad (1) & q_1a \rightarrow q_1aR & (2) & q_1B \rightarrow q_1BR & (3) & q_1b \rightarrow q_2BL & (4) \\ & q_2a \rightarrow q_2aL & (5) & q_2B \rightarrow q_2BL & (6) & q_2A \rightarrow q_0AR & (7) & q_0B \rightarrow q_3BR & (8) \\ & q_3B \rightarrow q_3BR & (9) & q_3\sqcup \rightarrow q_4\sqcup N & (10) \end{aligned}$$

- $M$  akzeptiert die Eingabe  $ab$  wie folgt:

$$q_0ab \underset{(1)}{\vdash} Aq_1b \underset{(4)}{\vdash} q_2AB \underset{(7)}{\vdash} Aq_0B \underset{(8)}{\vdash} ABq_3\sqcup \underset{(10)}{\vdash} ABq_4\sqcup$$

- Die MPCP-Instanz  $f(ab)$  enthält für  $u \in \Gamma$  die Wortpaare

Startregel	Kopierregeln	Löschregeln	Abschlussregel
$(\langle, \langle   z_0ab)$	$(u, u), ( ,  )$	$(q_4u, q_4), (uq_4, q_4)$	$(q_4  \rangle, \rangle)$

sowie folgende Überführungsregeln:

# Das Postsche Korrespondenzproblem

## Beispiel

Anweisung	zugehörige Überführungsregeln
$q_0 a \rightarrow q_1 AR$ (1)	$(q_0 a, Aq_1)$
$q_1 a \rightarrow q_1 aR$ (2)	$(q_1 a, aq_1)$
$q_1 B \rightarrow q_1 BR$ (3)	$(q_1 B, Bq_1)$
$q_1 b \rightarrow q_2 BL$ (4)	$(uq_1 b, q_2 uB), ( q_1 b,  q_2 \sqcup B)$
$q_2 a \rightarrow q_2 aL$ (5)	$(uq_2 a, q_2 ua), ( q_2 a,  q_2 \sqcup a)$
$q_2 B \rightarrow q_2 BL$ (6)	$(uq_2 B, q_2 uB), ( q_2 B,  q_2 \sqcup B)$
$q_2 A \rightarrow q_0 AR$ (7)	$(q_2 A, Aq_0)$
$q_0 B \rightarrow q_3 BR$ (8)	$(q_0 B, Bq_3)$
$q_3 B \rightarrow q_3 BR$ (9)	$(q_3 B, Bq_3)$
$q_3 \sqcup \rightarrow q_4 \sqcup N$ (10)	$(q_3 \sqcup, q_4 \sqcup), (q_3  , q_4 \sqcup  )$

## Beispiel

- Die MPCP-Instanz  $f(ab)$  enthält für  $u \in \Gamma$  die Wortpaare

Startregel	Kopierregeln	Löschregeln	Abschlussregel
$(\langle, \langle   z_0 ab)$	$(u, u), ( ,  )$	$(q_4 u, q_4), (u q_4, q_4)$	$(q_4   \rangle, \rangle)$

sowie u.a. folgende Überführungsregeln:

$q_0 a \rightarrow q_1 AR$	(1)	$(q_0 a, Aq_1)$
$q_1 b \rightarrow q_2 BL$	(4)	$(uq_1 b, q_2 uB), ( q_1 b,  q_2 \sqcup B)$
$q_2 A \rightarrow q_0 AR$	(7)	$(q_2 A, Aq_0)$
$q_0 B \rightarrow q_3 BR$	(8)	$(q_0 B, Bq_3)$
$q_3 \sqcup \rightarrow q_4 \sqcup N$	(10)	$(q_3 \sqcup, q_4 \sqcup), (q_3  , q_4 \sqcup  )$

- Der akzeptierenden Rechnung

$$q_0 ab \underset{(1)}{\vdash} Aq_1 b \underset{(4)}{\vdash} q_2 AB \underset{(7)}{\vdash} Aq_0 B \underset{(8)}{\vdash} ABq_3 \sqcup \underset{10}{\vdash} ABq_4 \sqcup$$

von  $M(ab)$  entspricht dann das MPCP-Lösungswort

## Beispiel

- Die MPCP-Instanz  $f(ab)$  enthält für  $u \in \Gamma$  die Wortpaare

Startregel	Kopierregeln	Löschregeln	Abschlussregel
$(\langle, \langle   z_0 ab)$	$(u, u), ( ,  )$	$(q_4 u, q_4), (u q_4, q_4)$	$(q_4   \rangle, \rangle)$

sowie u.a. folgende Überführungsregeln:

$q_0 a \rightarrow q_1 AR$	(1)	$(q_0 a, Aq_1)$
$q_1 b \rightarrow q_2 BL$	(4)	$(uq_1 b, q_2 uB), ( q_1 b,  q_2 \sqcup B)$
$q_2 A \rightarrow q_0 AR$	(7)	$(q_2 A, Aq_0)$
$q_0 B \rightarrow q_3 BR$	(8)	$(q_0 B, Bq_3)$
$q_3 \sqcup \rightarrow q_4 \sqcup N$	(10)	$(q_3 \sqcup, q_4 \sqcup), (q_3  , q_4 \sqcup  )$

- Der akzeptierenden Rechnung

$$q_0 ab \stackrel{(1)}{\vdash} Aq_1 b \stackrel{(4)}{\vdash} q_2 AB \stackrel{(7)}{\vdash} Aq_0 B \stackrel{(8)}{\vdash} ABq_3 \sqcup \stackrel{10}{\vdash} ABq_4 \sqcup$$

von  $M(ab)$  entspricht dann das MPCP-Lösungswort

$\langle$   
 $\langle | q_0 ab$

## Beispiel

- Die MPCP-Instanz  $f(ab)$  enthält für  $u \in \Gamma$  die Wortpaare

Startregel	Kopierregeln	Löschregeln	Abschlussregel
$(\langle, \langle   z_0 ab)$	$(u, u), ( ,  )$	$(q_4 u, q_4), (u q_4, q_4)$	$(q_4   \rangle, \rangle)$

sowie u.a. folgende Überführungsregeln:

$q_0 a \rightarrow q_1 AR$	(1)	$(q_0 a, Aq_1)$
$q_1 b \rightarrow q_2 BL$	(4)	$(uq_1 b, q_2 uB), ( q_1 b,  q_2 \sqcup B)$
$q_2 A \rightarrow q_0 AR$	(7)	$(q_2 A, Aq_0)$
$q_0 B \rightarrow q_3 BR$	(8)	$(q_0 B, Bq_3)$
$q_3 \sqcup \rightarrow q_4 \sqcup N$	(10)	$(q_3 \sqcup, q_4 \sqcup), (q_3  , q_4 \sqcup  )$

- Der akzeptierenden Rechnung

$$q_0 ab \stackrel{(1)}{\vdash} Aq_1 b \stackrel{(4)}{\vdash} q_2 AB \stackrel{(7)}{\vdash} Aq_0 B \stackrel{(8)}{\vdash} ABq_3 \sqcup \stackrel{10}{\vdash} ABq_4 \sqcup$$

von  $M(ab)$  entspricht dann das MPCP-Lösungswort

$\langle |$   
 $\langle | q_0 ab |$

## Beispiel

- Die MPCP-Instanz  $f(ab)$  enthält für  $u \in \Gamma$  die Wortpaare

Startregel	Kopierregeln	Löschregeln	Abschlussregel
$(\langle, \langle   z_0 ab)$	$(u, u), ( ,  )$	$(q_4 u, q_4), (u q_4, q_4)$	$(q_4   \rangle, \rangle)$

sowie u.a. folgende Überführungsregeln:

$q_0 a \rightarrow q_1 AR$	(1)	$(q_0 a, Aq_1)$
$q_1 b \rightarrow q_2 BL$	(4)	$(uq_1 b, q_2 uB), ( q_1 b,  q_2 \sqcup B)$
$q_2 A \rightarrow q_0 AR$	(7)	$(q_2 A, Aq_0)$
$q_0 B \rightarrow q_3 BR$	(8)	$(q_0 B, Bq_3)$
$q_3 \sqcup \rightarrow q_4 \sqcup N$	(10)	$(q_3 \sqcup, q_4 \sqcup), (q_3  , q_4 \sqcup  )$

- Der akzeptierenden Rechnung

$$q_0 ab \stackrel{(1)}{\vdash} Aq_1 b \stackrel{(4)}{\vdash} q_2 AB \stackrel{(7)}{\vdash} Aq_0 B \stackrel{(8)}{\vdash} ABq_3 \sqcup \stackrel{10}{\vdash} ABq_4 \sqcup$$

von  $M(ab)$  entspricht dann das MPCP-Lösungswort

$$\begin{array}{l} \langle | q_0 a \\ \langle | q_0 ab | Aq_1 \end{array}$$

## Beispiel

- Die MPCP-Instanz  $f(ab)$  enthält für  $u \in \Gamma$  die Wortpaare

Startregel	Kopierregeln	Löschregeln	Abschlussregel
$(\langle, \langle   z_0 ab)$	$(u, u), ( ,  )$	$(q_4 u, q_4), (u q_4, q_4)$	$(q_4   \rangle, \rangle)$

sowie u.a. folgende Überführungsregeln:

$q_0 a \rightarrow q_1 AR$	(1)	$(q_0 a, Aq_1)$
$q_1 b \rightarrow q_2 BL$	(4)	$(uq_1 b, q_2 uB), ( q_1 b,  q_2 \sqcup B)$
$q_2 A \rightarrow q_0 AR$	(7)	$(q_2 A, Aq_0)$
$q_0 B \rightarrow q_3 BR$	(8)	$(q_0 B, Bq_3)$
$q_3 \sqcup \rightarrow q_4 \sqcup N$	(10)	$(q_3 \sqcup, q_4 \sqcup), (q_3  , q_4 \sqcup  )$

- Der akzeptierenden Rechnung

$$q_0 ab \stackrel{(1)}{\vdash} Aq_1 b \stackrel{(4)}{\vdash} q_2 AB \stackrel{(7)}{\vdash} Aq_0 B \stackrel{(8)}{\vdash} ABq_3 \sqcup \stackrel{10}{\vdash} ABq_4 \sqcup$$

von  $M(ab)$  entspricht dann das MPCP-Lösungswort

$\langle | q_0 ab$   
 $\langle | q_0 ab | Aq_1 b$



## Beispiel

- Die MPCP-Instanz  $f(ab)$  enthält für  $u \in \Gamma$  die Wortpaare

Startregel	Kopierregeln	Löschregeln	Abschlussregel
$(\langle, \langle   z_0 ab)$	$(u, u), ( ,  )$	$(q_4 u, q_4), (u q_4, q_4)$	$(q_4   \rangle, \rangle)$

sowie u.a. folgende Überführungsregeln:

$q_0 a \rightarrow q_1 AR$	(1)	$(q_0 a, Aq_1)$
$q_1 b \rightarrow q_2 BL$	(4)	$(uq_1 b, q_2 uB), ( q_1 b,  q_2 \sqcup B)$
$q_2 A \rightarrow q_0 AR$	(7)	$(q_2 A, Aq_0)$
$q_0 B \rightarrow q_3 BR$	(8)	$(q_0 B, Bq_3)$
$q_3 \sqcup \rightarrow q_4 \sqcup N$	(10)	$(q_3 \sqcup, q_4 \sqcup), (q_3  , q_4 \sqcup  )$

- Der akzeptierenden Rechnung

$$q_0 ab \stackrel{(1)}{\vdash} Aq_1 b \stackrel{(4)}{\vdash} q_2 AB \stackrel{(7)}{\vdash} Aq_0 B \stackrel{(8)}{\vdash} ABq_3 \sqcup \stackrel{10}{\vdash} ABq_4 \sqcup$$

von  $M(ab)$  entspricht dann das MPCP-Lösungswort

$$\begin{aligned} &\langle | q_0 ab | \\ &\langle | q_0 ab | Aq_1 b | \end{aligned}$$

## Beispiel

- Die MPCP-Instanz  $f(ab)$  enthält für  $u \in \Gamma$  die Wortpaare

Startregel	Kopierregeln	Löschregeln	Abschlussregel
$(\langle, \langle   z_0 ab)$	$(u, u), ( ,  )$	$(q_4 u, q_4), (u q_4, q_4)$	$(q_4   \rangle, \rangle)$

sowie u.a. folgende Überführungsregeln:

$q_0 a \rightarrow q_1 A R$	(1)	$(q_0 a, A q_1)$
$q_1 b \rightarrow q_2 B L$	(4)	$(u q_1 b, q_2 u B), (  q_1 b,   q_2 \sqcup B)$
$q_2 A \rightarrow q_0 A R$	(7)	$(q_2 A, A q_0)$
$q_0 B \rightarrow q_3 B R$	(8)	$(q_0 B, B q_3)$
$q_3 \sqcup \rightarrow q_4 \sqcup N$	(10)	$(q_3 \sqcup, q_4 \sqcup), (q_3  , q_4 \sqcup  )$

- Der akzeptierenden Rechnung

$$q_0 ab \stackrel{(1)}{\vdash} A q_1 b \stackrel{(4)}{\vdash} q_2 AB \stackrel{(7)}{\vdash} A q_0 B \stackrel{(8)}{\vdash} AB q_3 \sqcup \stackrel{10}{\vdash} AB q_4 \sqcup$$

von  $M(ab)$  entspricht dann das MPCP-Lösungswort

$$\begin{aligned} &\langle | q_0 ab | A q_1 b \\ &\langle | q_0 ab | A q_1 b | q_2 AB \end{aligned}$$

## Beispiel

- Die MPCP-Instanz  $f(ab)$  enthält für  $u \in \Gamma$  die Wortpaare

Startregel	Kopierregeln	Löschregeln	Abschlussregel
$(\langle, \langle   z_0 ab)$	$(u, u), ( ,  )$	$(q_4 u, q_4), (u q_4, q_4)$	$(q_4   \rangle, \rangle)$

sowie u.a. folgende Überführungsregeln:

$q_0 a \rightarrow q_1 AR$	(1)	$(q_0 a, Aq_1)$
$q_1 b \rightarrow q_2 BL$	(4)	$(uq_1 b, q_2 uB), ( q_1 b,  q_2 \sqcup B)$
$q_2 A \rightarrow q_0 AR$	(7)	$(q_2 A, Aq_0)$
$q_0 B \rightarrow q_3 BR$	(8)	$(q_0 B, Bq_3)$
$q_3 \sqcup \rightarrow q_4 \sqcup N$	(10)	$(q_3 \sqcup, q_4 \sqcup), (q_3  , q_4 \sqcup  )$

- Der akzeptierenden Rechnung

$$q_0 ab \underset{(1)}{\vdash} Aq_1 b \underset{(4)}{\vdash} q_2 AB \underset{(7)}{\vdash} Aq_0 B \underset{(8)}{\vdash} ABq_3 \sqcup \underset{10}{\vdash} ABq_4 \sqcup$$

von  $M(ab)$  entspricht dann das MPCP-Lösungswort

$$\begin{aligned} &\langle | q_0 ab | Aq_1 b | \\ &\langle | q_0 ab | Aq_1 b | q_2 AB | \end{aligned}$$

## Beispiel

- Die MPCP-Instanz  $f(ab)$  enthält für  $u \in \Gamma$  die Wortpaare

Startregel	Kopierregeln	Löschregeln	Abschlussregel
$(\langle, \langle   z_0 ab)$	$(u, u), ( ,  )$	$(q_4 u, q_4), (u q_4, q_4)$	$(q_4   \rangle, \rangle)$

sowie u.a. folgende Überführungsregeln:

$q_0 a \rightarrow q_1 A R$	(1)	$(q_0 a, A q_1)$
$q_1 b \rightarrow q_2 B L$	(4)	$(u q_1 b, q_2 u B), (  q_1 b,   q_2 \sqcup B)$
$q_2 A \rightarrow q_0 A R$	(7)	$(q_2 A, A q_0)$
$q_0 B \rightarrow q_3 B R$	(8)	$(q_0 B, B q_3)$
$q_3 \sqcup \rightarrow q_4 \sqcup N$	(10)	$(q_3 \sqcup, q_4 \sqcup), (q_3  , q_4 \sqcup  )$

- Der akzeptierenden Rechnung

$$q_0 ab \stackrel{(1)}{\vdash} A q_1 b \stackrel{(4)}{\vdash} q_2 AB \stackrel{(7)}{\vdash} A q_0 B \stackrel{(8)}{\vdash} AB q_3 \sqcup \stackrel{10}{\vdash} AB q_4 \sqcup$$

von  $M(ab)$  entspricht dann das MPCP-Lösungswort

$$\begin{aligned} &\langle | q_0 ab | A q_1 b | q_2 A \\ &\langle | q_0 ab | A q_1 b | q_2 AB | A q_0 \end{aligned}$$

## Beispiel

- Die MPCP-Instanz  $f(ab)$  enthält für  $u \in \Gamma$  die Wortpaare

Startregel	Kopierregeln	Löschregeln	Abschlussregel
$(\langle, \langle   z_0 ab)$	$(u, u), ( ,  )$	$(q_4 u, q_4), (uq_4, q_4)$	$(q_4   \rangle, \rangle)$

sowie u.a. folgende Überführungsregeln:

$q_0 a \rightarrow q_1 AR$	(1)	$(q_0 a, Aq_1)$
$q_1 b \rightarrow q_2 BL$	(4)	$(uq_1 b, q_2 uB), ( q_1 b,  q_2 \sqcup B)$
$q_2 A \rightarrow q_0 AR$	(7)	$(q_2 A, Aq_0)$
$q_0 B \rightarrow q_3 BR$	(8)	$(q_0 B, Bq_3)$
$q_3 \sqcup \rightarrow q_4 \sqcup N$	(10)	$(q_3 \sqcup, q_4 \sqcup), (q_3  , q_4 \sqcup  )$

- Der akzeptierenden Rechnung

$$q_0 ab \stackrel{(1)}{\vdash} Aq_1 b \stackrel{(4)}{\vdash} q_2 AB \stackrel{(7)}{\vdash} Aq_0 B \stackrel{(8)}{\vdash} ABq_3 \sqcup \stackrel{10}{\vdash} ABq_4 \sqcup$$

von  $M(ab)$  entspricht dann das MPCP-Lösungswort

$$\begin{aligned} &\langle | q_0 ab | Aq_1 b | q_2 AB \\ &\langle | q_0 ab | Aq_1 b | q_2 AB | Aq_0 B \end{aligned}$$

- Die MPCP-Instanz  $f(ab)$  enthält für  $u \in \Gamma$  die Wortpaare

Startregel	Kopierregeln	Löschregeln	Abschlussregel
$(\langle, \langle   z_0 ab)$	$(u, u), ( ,  )$	$(q_4 u, q_4), (u q_4, q_4)$	$(q_4   \rangle, \rangle)$

sowie u.a. folgende Überführungsregeln:

$q_0 a \rightarrow q_1 AR$	(1)	$(q_0 a, Aq_1)$
$q_1 b \rightarrow q_2 BL$	(4)	$(uq_1 b, q_2 uB), ( q_1 b,  q_2 \sqcup B)$
$q_2 A \rightarrow q_0 AR$	(7)	$(q_2 A, Aq_0)$
$q_0 B \rightarrow q_3 BR$	(8)	$(q_0 B, Bq_3)$
$q_3 \sqcup \rightarrow q_4 \sqcup N$	(10)	$(q_3 \sqcup, q_4 \sqcup), (q_3  , q_4 \sqcup  )$

- Der akzeptierenden Rechnung

$$q_0 ab \stackrel{(1)}{\vdash} Aq_1 b \stackrel{(4)}{\vdash} q_2 AB \stackrel{(7)}{\vdash} Aq_0 B \stackrel{(8)}{\vdash} ABq_3 \sqcup \stackrel{10}{\vdash} ABq_4 \sqcup$$

von  $M(ab)$  entspricht dann das MPCP-Lösungswort

$$\begin{aligned} &\langle | q_0 ab | Aq_1 b | q_2 AB | \\ &\langle | q_0 ab | Aq_1 b | q_2 AB | Aq_0 B | \end{aligned}$$

## Beispiel

- Die MPCP-Instanz  $f(ab)$  enthält für  $u \in \Gamma$  die Wortpaare

Startregel	Kopierregeln	Löschregeln	Abschlussregel
$(\langle, \langle   z_0 ab)$	$(u, u), ( ,  )$	$(q_4 u, q_4), (uq_4, q_4)$	$(q_4   \rangle, \rangle)$

sowie u.a. folgende Überführungsregeln:

$q_0 a \rightarrow q_1 AR$	(1)	$(q_0 a, Aq_1)$
$q_1 b \rightarrow q_2 BL$	(4)	$(uq_1 b, q_2 uB), ( q_1 b,  q_2 \sqcup B)$
$q_2 A \rightarrow q_0 AR$	(7)	$(q_2 A, Aq_0)$
$q_0 B \rightarrow q_3 BR$	(8)	$(q_0 B, Bq_3)$
$q_3 \sqcup \rightarrow q_4 \sqcup N$	(10)	$(q_3 \sqcup, q_4 \sqcup), (q_3  , q_4 \sqcup  )$

- Der akzeptierenden Rechnung

$$q_0 ab \underset{(1)}{\vdash} Aq_1 b \underset{(4)}{\vdash} q_2 AB \underset{(7)}{\vdash} Aq_0 B \underset{(8)}{\vdash} ABq_3 \sqcup \underset{10}{\vdash} ABq_4 \sqcup$$

von  $M(ab)$  entspricht dann das MPCP-Lösungswort

$$\begin{aligned} &\langle | q_0 ab | Aq_1 b | q_2 AB | A \\ &\langle | q_0 ab | Aq_1 b | q_2 AB | Aq_0 B | A \end{aligned}$$

## Beispiel

- Die MPCP-Instanz  $f(ab)$  enthält für  $u \in \Gamma$  die Wortpaare

Startregel	Kopierregeln	Löschregeln	Abschlussregel
$(\langle, \langle   z_0 ab)$	$(u, u), ( ,  )$	$(q_4 u, q_4), (u q_4, q_4)$	$(q_4   \rangle, \rangle)$

sowie u.a. folgende Überführungsregeln:

$q_0 a \rightarrow q_1 AR$	(1)	$(q_0 a, Aq_1)$
$q_1 b \rightarrow q_2 BL$	(4)	$(uq_1 b, q_2 uB), ( q_1 b,  q_2 \sqcup B)$
$q_2 A \rightarrow q_0 AR$	(7)	$(q_2 A, Aq_0)$
$q_0 B \rightarrow q_3 BR$	(8)	$(q_0 B, Bq_3)$
$q_3 \sqcup \rightarrow q_4 \sqcup N$	(10)	$(q_3 \sqcup, q_4 \sqcup), (q_3  , q_4 \sqcup  )$

- Der akzeptierenden Rechnung

$$q_0 ab \stackrel{(1)}{\vdash} Aq_1 b \stackrel{(4)}{\vdash} q_2 AB \stackrel{(7)}{\vdash} Aq_0 B \stackrel{(8)}{\vdash} ABq_3 \stackrel{10}{\vdash} ABq_4 \sqcup$$

von  $M(ab)$  entspricht dann das MPCP-Lösungswort

$$\begin{aligned} &\langle | q_0 ab | Aq_1 b | q_2 AB | Aq_0 B \\ &\langle | q_0 ab | Aq_1 b | q_2 AB | Aq_0 B | ABq_3 \end{aligned}$$



## Beispiel

- Die MPCP-Instanz  $f(ab)$  enthält für  $u \in \Gamma$  die Wortpaare

Startregel	Kopierregeln	Löschregeln	Abschlussregel
$(\langle, \langle   z_0 ab)$	$(u, u), ( ,  )$	$(q_4 u, q_4), (u q_4, q_4)$	$(q_4   \rangle, \rangle)$

sowie u.a. folgende Überführungsregeln:

$q_0 a \rightarrow q_1 AR$	(1)	$(q_0 a, Aq_1)$
$q_1 b \rightarrow q_2 BL$	(4)	$(uq_1 b, q_2 uB), ( q_1 b,  q_2 \sqcup B)$
$q_2 A \rightarrow q_0 AR$	(7)	$(q_2 A, Aq_0)$
$q_0 B \rightarrow q_3 BR$	(8)	$(q_0 B, Bq_3)$
$q_3 \sqcup \rightarrow q_4 \sqcup N$	(10)	$(q_3 \sqcup, q_4 \sqcup), (q_3  , q_4 \sqcup  )$

- Der akzeptierenden Rechnung

$$q_0 ab \underset{(1)}{\vdash} Aq_1 b \underset{(4)}{\vdash} q_2 AB \underset{(7)}{\vdash} Aq_0 B \underset{(8)}{\vdash} ABq_3 \sqcup \underset{10}{\vdash} ABq_4 \sqcup$$

von  $M(ab)$  entspricht dann das MPCP-Lösungswort

$$\begin{aligned} &\langle | q_0 ab | Aq_1 b | q_2 AB | Aq_0 B | \\ &\langle | q_0 ab | Aq_1 b | q_2 AB | Aq_0 B | ABq_3 | \end{aligned}$$

## Beispiel

- Die MPCP-Instanz  $f(ab)$  enthält für  $u \in \Gamma$  die Wortpaare

Startregel	Kopierregeln	Löschregeln	Abschlussregel
$(\langle, \langle   z_0 ab)$	$(u, u), ( ,  )$	$(q_4 u, q_4), (uq_4, q_4)$	$(q_4   \rangle, \rangle)$

sowie u.a. folgende Überführungsregeln:

$q_0 a \rightarrow q_1 AR$	(1)	$(q_0 a, Aq_1)$
$q_1 b \rightarrow q_2 BL$	(4)	$(uq_1 b, q_2 uB), ( q_1 b,  q_2 \sqcup B)$
$q_2 A \rightarrow q_0 AR$	(7)	$(q_2 A, Aq_0)$
$q_0 B \rightarrow q_3 BR$	(8)	$(q_0 B, Bq_3)$
$q_3 \sqcup \rightarrow q_4 \sqcup N$	(10)	$(q_3 \sqcup, q_4 \sqcup), (q_3  , q_4 \sqcup  )$

- Der akzeptierenden Rechnung

$$q_0 ab \underset{(1)}{\vdash} Aq_1 b \underset{(4)}{\vdash} q_2 AB \underset{(7)}{\vdash} Aq_0 B \underset{(8)}{\vdash} ABq_3 \sqcup \underset{10}{\vdash} ABq_4 \sqcup$$

von  $M(ab)$  entspricht dann das MPCP-Lösungswort

$$\begin{aligned} &\langle | q_0 ab | Aq_1 b | q_2 AB | Aq_0 B | A \\ &\langle | q_0 ab | Aq_1 b | q_2 AB | Aq_0 B | ABq_3 | A \end{aligned}$$

## Beispiel

- Die MPCP-Instanz  $f(ab)$  enthält für  $u \in \Gamma$  die Wortpaare

Startregel	Kopierregeln	Löschregeln	Abschlussregel
$(\langle, \langle   z_0 ab)$	$(u, u), ( ,  )$	$(q_4 u, q_4), (u q_4, q_4)$	$(q_4   \rangle, \rangle)$

sowie u.a. folgende Überführungsregeln:

$q_0 a \rightarrow q_1 AR$	(1)	$(q_0 a, Aq_1)$
$q_1 b \rightarrow q_2 BL$	(4)	$(uq_1 b, q_2 uB), ( q_1 b,  q_2 \sqcup B)$
$q_2 A \rightarrow q_0 AR$	(7)	$(q_2 A, Aq_0)$
$q_0 B \rightarrow q_3 BR$	(8)	$(q_0 B, Bq_3)$
$q_3 \sqcup \rightarrow q_4 \sqcup N$	(10)	$(q_3 \sqcup, q_4 \sqcup), (q_3  , q_4 \sqcup  )$

- Der akzeptierenden Rechnung

$$q_0 ab \stackrel{(1)}{\vdash} Aq_1 b \stackrel{(4)}{\vdash} q_2 AB \stackrel{(7)}{\vdash} Aq_0 B \stackrel{(8)}{\vdash} ABq_3 \sqcup \stackrel{10}{\vdash} ABq_4 \sqcup$$

von  $M(ab)$  entspricht dann das MPCP-Lösungswort

$$\begin{aligned} &\langle | q_0 ab | Aq_1 b | q_2 AB | Aq_0 B | AB \\ &\langle | q_0 ab | Aq_1 b | q_2 AB | Aq_0 B | ABq_3 | AB \end{aligned}$$

## Beispiel

- Die MPCP-Instanz  $f(ab)$  enthält für  $u \in \Gamma$  die Wortpaare

Startregel	Kopierregeln	Löschregeln	Abschlussregel
$(\langle, \langle   z_0 ab)$	$(u, u), ( ,  )$	$(q_4 u, q_4), (u q_4, q_4)$	$(q_4   \rangle, \rangle)$

sowie u.a. folgende Überführungsregeln:

$q_0 a \rightarrow q_1 AR$	(1)	$(q_0 a, Aq_1)$
$q_1 b \rightarrow q_2 BL$	(4)	$(uq_1 b, q_2 uB), ( q_1 b,  q_2 \sqcup B)$
$q_2 A \rightarrow q_0 AR$	(7)	$(q_2 A, Aq_0)$
$q_0 B \rightarrow q_3 BR$	(8)	$(q_0 B, Bq_3)$
$q_3 \sqcup \rightarrow q_4 \sqcup N$	(10)	$(q_3 \sqcup, q_4 \sqcup), (q_3  , q_4 \sqcup  )$

- Der akzeptierenden Rechnung

$$q_0 ab \underset{(1)}{\vdash} Aq_1 b \underset{(4)}{\vdash} q_2 AB \underset{(7)}{\vdash} Aq_0 B \underset{(8)}{\vdash} ABq_3 \sqcup \underset{10}{\vdash} ABq_4 \sqcup$$

von  $M(ab)$  entspricht dann das MPCP-Lösungswort

$$\begin{aligned} &\langle | q_0 ab | Aq_1 b | q_2 AB | Aq_0 B | ABq_3 | \\ &\langle | q_0 ab | Aq_1 b | q_2 AB | Aq_0 B | ABq_3 | ABq_4 \sqcup | \end{aligned}$$

## Beispiel

- Die MPCP-Instanz  $f(ab)$  enthält für  $u \in \Gamma$  die Wortpaare

Startregel	Kopierregeln	Löschregeln	Abschlussregel
$(\langle, \langle   z_0 ab)$	$(u, u), ( ,  )$	$(q_4 u, q_4), (uq_4, q_4)$	$(q_4   \rangle, \rangle)$

sowie u.a. folgende Überführungsregeln:

$q_0 a \rightarrow q_1 AR$	(1)	$(q_0 a, Aq_1)$
$q_1 b \rightarrow q_2 BL$	(4)	$(uq_1 b, q_2 uB), ( q_1 b,  q_2 \sqcup B)$
$q_2 A \rightarrow q_0 AR$	(7)	$(q_2 A, Aq_0)$
$q_0 B \rightarrow q_3 BR$	(8)	$(q_0 B, Bq_3)$
$q_3 \sqcup \rightarrow q_4 \sqcup N$	(10)	$(q_3 \sqcup, q_4 \sqcup), (q_3  , q_4 \sqcup  )$

- Der akzeptierenden Rechnung

$$q_0 ab \stackrel{(1)}{\vdash} Aq_1 b \stackrel{(4)}{\vdash} q_2 AB \stackrel{(7)}{\vdash} Aq_0 B \stackrel{(8)}{\vdash} ABq_3 \stackrel{10}{\vdash} ABq_4 \sqcup$$

von  $M(ab)$  entspricht dann das MPCP-Lösungswort

$\langle | q_0 ab | Aq_1 b | q_2 AB | Aq_0 B | ABq_3 | A$

$\langle | q_0 ab | Aq_1 b | q_2 AB | Aq_0 B | ABq_3 | ABq_4 \sqcup | A$

## Beispiel

- Die MPCP-Instanz  $f(ab)$  enthält für  $u \in \Gamma$  die Wortpaare

Startregel	Kopierregeln	Löschregeln	Abschlussregel
$(\langle, \langle   z_0 ab)$	$(u, u), ( ,  )$	$(q_4 u, q_4), (uq_4, q_4)$	$(q_4   \rangle, \rangle)$

sowie u.a. folgende Überführungsregeln:

$q_0 a \rightarrow q_1 AR$	(1)	$(q_0 a, Aq_1)$
$q_1 b \rightarrow q_2 BL$	(4)	$(uq_1 b, q_2 uB), ( q_1 b,  q_2 \sqcup B)$
$q_2 A \rightarrow q_0 AR$	(7)	$(q_2 A, Aq_0)$
$q_0 B \rightarrow q_3 BR$	(8)	$(q_0 B, Bq_3)$
$q_3 \sqcup \rightarrow q_4 \sqcup N$	(10)	$(q_3 \sqcup, q_4 \sqcup), (q_3  , q_4 \sqcup  )$

- Der akzeptierenden Rechnung

$$q_0 ab \underset{(1)}{\vdash} Aq_1 b \underset{(4)}{\vdash} q_2 AB \underset{(7)}{\vdash} Aq_0 B \underset{(8)}{\vdash} ABq_3 \underset{10}{\vdash} ABq_4 \sqcup$$

von  $M(ab)$  entspricht dann das MPCP-Lösungswort

$$\begin{aligned} &\langle | q_0 ab | Aq_1 b | q_2 AB | Aq_0 B | ABq_3 | ABq_4 \\ &\langle | q_0 ab | Aq_1 b | q_2 AB | Aq_0 B | ABq_3 | ABq_4 \sqcup | Aq_4 \end{aligned}$$

## Beispiel

- Die MPCP-Instanz  $f(ab)$  enthält für  $u \in \Gamma$  die Wortpaare

Startregel	Kopierregeln	Löschregeln	Abschlussregel
$(\langle, \langle   z_0 ab)$	$(u, u), ( ,  )$	$(q_4 u, q_4), (u q_4, q_4)$	$(q_4   \rangle, \rangle)$

sowie u.a. folgende Überführungsregeln:

$q_0 a \rightarrow q_1 AR$	(1)	$(q_0 a, Aq_1)$
$q_1 b \rightarrow q_2 BL$	(4)	$(uq_1 b, q_2 uB), ( q_1 b,  q_2 \sqcup B)$
$q_2 A \rightarrow q_0 AR$	(7)	$(q_2 A, Aq_0)$
$q_0 B \rightarrow q_3 BR$	(8)	$(q_0 B, Bq_3)$
$q_3 \sqcup \rightarrow q_4 \sqcup N$	(10)	$(q_3 \sqcup, q_4 \sqcup), (q_3  , q_4 \sqcup  )$

- Der akzeptierenden Rechnung

$$q_0 ab \underset{(1)}{\vdash} Aq_1 b \underset{(4)}{\vdash} q_2 AB \underset{(7)}{\vdash} Aq_0 B \underset{(8)}{\vdash} ABq_3 \underset{10}{\vdash} ABq_4 \sqcup$$

von  $M(ab)$  entspricht dann das MPCP-Lösungswort

$\langle | q_0 ab | Aq_1 b | q_2 AB | Aq_0 B | ABq_3 | ABq_4 \sqcup$

$\langle | q_0 ab | Aq_1 b | q_2 AB | Aq_0 B | ABq_3 | ABq_4 \sqcup | Aq_4 \sqcup$

## Beispiel

- Die MPCP-Instanz  $f(ab)$  enthält für  $u \in \Gamma$  die Wortpaare

Startregel	Kopierregeln	Löschregeln	Abschlussregel
$(\langle, \langle   z_0 ab)$	$(u, u), ( ,  )$	$(q_4 u, q_4), (u q_4, q_4)$	$(q_4   \rangle, \rangle)$

sowie u.a. folgende Überführungsregeln:

$q_0 a \rightarrow q_1 AR$	(1)	$(q_0 a, Aq_1)$
$q_1 b \rightarrow q_2 BL$	(4)	$(uq_1 b, q_2 uB), ( q_1 b,  q_2 \sqcup B)$
$q_2 A \rightarrow q_0 AR$	(7)	$(q_2 A, Aq_0)$
$q_0 B \rightarrow q_3 BR$	(8)	$(q_0 B, Bq_3)$
$q_3 \sqcup \rightarrow q_4 \sqcup N$	(10)	$(q_3 \sqcup, q_4 \sqcup), (q_3  , q_4 \sqcup  )$

- Der akzeptierenden Rechnung

$$q_0 ab \underset{(1)}{\vdash} Aq_1 b \underset{(4)}{\vdash} q_2 AB \underset{(7)}{\vdash} Aq_0 B \underset{(8)}{\vdash} ABq_3 \underset{10}{\vdash} ABq_4 \sqcup$$

von  $M(ab)$  entspricht dann das MPCP-Lösungswort

$$\begin{aligned} &\langle | q_0 ab | Aq_1 b | q_2 AB | Aq_0 B | ABq_3 | ABq_4 \sqcup | \\ &\langle | q_0 ab | Aq_1 b | q_2 AB | Aq_0 B | ABq_3 | ABq_4 \sqcup | Aq_4 \sqcup | \end{aligned}$$



## Beispiel

- Die MPCP-Instanz  $f(ab)$  enthält für  $u \in \Gamma$  die Wortpaare

Startregel	Kopierregeln	Löschregeln	Abschlussregel
$(\langle, \langle   z_0 ab)$	$(u, u), ( ,  )$	$(q_4 u, q_4), (uq_4, q_4)$	$(q_4   \rangle, \rangle)$

sowie u.a. folgende Überführungsregeln:

$q_0 a \rightarrow q_1 AR$	(1)	$(q_0 a, Aq_1)$
$q_1 b \rightarrow q_2 BL$	(4)	$(uq_1 b, q_2 uB), ( q_1 b,  q_2 \sqcup B)$
$q_2 A \rightarrow q_0 AR$	(7)	$(q_2 A, Aq_0)$
$q_0 B \rightarrow q_3 BR$	(8)	$(q_0 B, Bq_3)$
$q_3 \sqcup \rightarrow q_4 \sqcup N$	(10)	$(q_3 \sqcup, q_4 \sqcup), (q_3  , q_4 \sqcup  )$

- Der akzeptierenden Rechnung

$$q_0 ab \stackrel{(1)}{\vdash} Aq_1 b \stackrel{(4)}{\vdash} q_2 AB \stackrel{(7)}{\vdash} Aq_0 B \stackrel{(8)}{\vdash} ABq_3 \stackrel{10}{\vdash} ABq_4 \sqcup$$

von  $M(ab)$  entspricht dann das MPCP-Lösungswort

$$\begin{aligned} &\langle | q_0 ab | Aq_1 b | q_2 AB | Aq_0 B | ABq_3 | ABq_4 \sqcup | Aq_4 \\ &\langle | q_0 ab | Aq_1 b | q_2 AB | Aq_0 B | ABq_3 | ABq_4 \sqcup | Aq_4 \sqcup | q_4 \end{aligned}$$

## Beispiel

- Die MPCP-Instanz  $f(ab)$  enthält für  $u \in \Gamma$  die Wortpaare

Startregel	Kopierregeln	Löschregeln	Abschlussregel
$(\langle, \langle   z_0 ab)$	$(u, u), ( ,  )$	$(q_4 u, q_4), (u q_4, q_4)$	$(q_4   \rangle, \rangle)$

sowie u.a. folgende Überführungsregeln:

$q_0 a \rightarrow q_1 AR$	(1)	$(q_0 a, Aq_1)$
$q_1 b \rightarrow q_2 BL$	(4)	$(uq_1 b, q_2 uB), ( q_1 b,  q_2 \sqcup B)$
$q_2 A \rightarrow q_0 AR$	(7)	$(q_2 A, Aq_0)$
$q_0 B \rightarrow q_3 BR$	(8)	$(q_0 B, Bq_3)$
$q_3 \sqcup \rightarrow q_4 \sqcup N$	(10)	$(q_3 \sqcup, q_4 \sqcup), (q_3  , q_4 \sqcup  )$

- Der akzeptierenden Rechnung

$$q_0 ab \stackrel{(1)}{\vdash} Aq_1 b \stackrel{(4)}{\vdash} q_2 AB \stackrel{(7)}{\vdash} Aq_0 B \stackrel{(8)}{\vdash} ABq_3 \stackrel{10}{\vdash} ABq_4 \sqcup$$

von  $M(ab)$  entspricht dann das MPCP-Lösungswort

$$\begin{aligned} &\langle | q_0 ab | Aq_1 b | q_2 AB | Aq_0 B | ABq_3 | ABq_4 \sqcup | Aq_4 \sqcup \\ &\langle | q_0 ab | Aq_1 b | q_2 AB | Aq_0 B | ABq_3 | ABq_4 \sqcup | Aq_4 \sqcup | q_4 \sqcup \end{aligned}$$

## Beispiel

- Die MPCP-Instanz  $f(ab)$  enthält für  $u \in \Gamma$  die Wortpaare

Startregel	Kopierregeln	Löschregeln	Abschlussregel
$(\langle, \langle   z_0 ab)$	$(u, u), ( ,  )$	$(q_4 u, q_4), (u q_4, q_4)$	$(q_4   \rangle, \rangle)$

sowie u.a. folgende Überführungsregeln:

$q_0 a \rightarrow q_1 AR$	(1)	$(q_0 a, Aq_1)$
$q_1 b \rightarrow q_2 BL$	(4)	$(uq_1 b, q_2 uB), ( q_1 b,  q_2 \sqcup B)$
$q_2 A \rightarrow q_0 AR$	(7)	$(q_2 A, Aq_0)$
$q_0 B \rightarrow q_3 BR$	(8)	$(q_0 B, Bq_3)$
$q_3 \sqcup \rightarrow q_4 \sqcup N$	(10)	$(q_3 \sqcup, q_4 \sqcup), (q_3  , q_4 \sqcup  )$

- Der akzeptierenden Rechnung

$$q_0 ab \stackrel{(1)}{\vdash} Aq_1 b \stackrel{(4)}{\vdash} q_2 AB \stackrel{(7)}{\vdash} Aq_0 B \stackrel{(8)}{\vdash} ABq_3 \sqcup \stackrel{10}{\vdash} ABq_4 \sqcup$$

von  $M(ab)$  entspricht dann das MPCP-Lösungswort

$$\begin{aligned} &\langle | q_0 ab | Aq_1 b | q_2 AB | Aq_0 B | ABq_3 | ABq_4 \sqcup | Aq_4 \sqcup | \\ &\langle | q_0 ab | Aq_1 b | q_2 AB | Aq_0 B | ABq_3 | ABq_4 \sqcup | Aq_4 \sqcup | q_4 \sqcup | \end{aligned}$$

## Beispiel

- Die MPCP-Instanz  $f(ab)$  enthält für  $u \in \Gamma$  die Wortpaare

Startregel	Kopierregeln	Löschregeln	Abschlussregel
$(\langle, \langle   z_0 ab)$	$(u, u), ( ,  )$	$(q_4 u, q_4), (u q_4, q_4)$	$(q_4   \rangle, \rangle)$

sowie u.a. folgende Überführungsregeln:

$q_0 a \rightarrow q_1 AR$	(1)	$(q_0 a, Aq_1)$
$q_1 b \rightarrow q_2 BL$	(4)	$(uq_1 b, q_2 uB), ( q_1 b,  q_2 \sqcup B)$
$q_2 A \rightarrow q_0 AR$	(7)	$(q_2 A, Aq_0)$
$q_0 B \rightarrow q_3 BR$	(8)	$(q_0 B, Bq_3)$
$q_3 \sqcup \rightarrow q_4 \sqcup N$	(10)	$(q_3 \sqcup, q_4 \sqcup), (q_3  , q_4 \sqcup  )$

- Der akzeptierenden Rechnung

$$q_0 ab \stackrel{(1)}{\vdash} Aq_1 b \stackrel{(4)}{\vdash} q_2 AB \stackrel{(7)}{\vdash} Aq_0 B \stackrel{(8)}{\vdash} ABq_3 \stackrel{10}{\vdash} ABq_4 \sqcup$$

von  $M(ab)$  entspricht dann das MPCP-Lösungswort

$$\begin{aligned} &\langle | q_0 ab | Aq_1 b | q_2 AB | Aq_0 B | ABq_3 | ABq_4 \sqcup | Aq_4 \sqcup | q_4 \sqcup \\ &\langle | q_0 ab | Aq_1 b | q_2 AB | Aq_0 B | ABq_3 | ABq_4 \sqcup | Aq_4 \sqcup | q_4 \sqcup | q_4 \end{aligned}$$

## Beispiel

- Die MPCP-Instanz  $f(ab)$  enthält für  $u \in \Gamma$  die Wortpaare

Startregel	Kopierregeln	Löschregeln	Abschlussregel
$(\langle, \langle   z_0 ab)$	$(u, u), ( ,  )$	$(q_4 u, q_4), (u q_4, q_4)$	$(q_4   \rangle, \rangle)$

sowie u.a. folgende Überführungsregeln:

$q_0 a \rightarrow q_1 AR$	(1)	$(q_0 a, Aq_1)$
$q_1 b \rightarrow q_2 BL$	(4)	$(uq_1 b, q_2 uB), ( q_1 b,  q_2 \sqcup B)$
$q_2 A \rightarrow q_0 AR$	(7)	$(q_2 A, Aq_0)$
$q_0 B \rightarrow q_3 BR$	(8)	$(q_0 B, Bq_3)$
$q_3 \sqcup \rightarrow q_4 \sqcup N$	(10)	$(q_3 \sqcup, q_4 \sqcup), (q_3  , q_4 \sqcup  )$

- Der akzeptierenden Rechnung

$$q_0 ab \stackrel{(1)}{\vdash} Aq_1 b \stackrel{(4)}{\vdash} q_2 AB \stackrel{(7)}{\vdash} Aq_0 B \stackrel{(8)}{\vdash} ABq_3 \stackrel{10}{\vdash} ABq_4 \sqcup$$

von  $M(ab)$  entspricht dann das MPCP-Lösungswort

$$\begin{aligned} &\langle | q_0 ab | Aq_1 b | q_2 AB | Aq_0 B | ABq_3 | ABq_4 \sqcup | Aq_4 \sqcup | q_4 \sqcup | \\ &\langle | q_0 ab | Aq_1 b | q_2 AB | Aq_0 B | ABq_3 | ABq_4 \sqcup | Aq_4 \sqcup | q_4 \sqcup | q_4 | \end{aligned}$$

- Die MPCP-Instanz  $f(ab)$  enthält für  $u \in \Gamma$  die Wortpaare

Startregel	Kopierregeln	Löschregeln	Abschlussregel
$(\langle, \langle   z_0 ab)$	$(u, u), ( ,  )$	$(q_4 u, q_4), (u q_4, q_4)$	$(q_4   \rangle, \rangle)$

sowie u.a. folgende Überführungsregeln:

$q_0 a \rightarrow q_1 AR$	(1)	$(q_0 a, Aq_1)$
$q_1 b \rightarrow q_2 BL$	(4)	$(uq_1 b, q_2 uB), ( q_1 b,  q_2 \sqcup B)$
$q_2 A \rightarrow q_0 AR$	(7)	$(q_2 A, Aq_0)$
$q_0 B \rightarrow q_3 BR$	(8)	$(q_0 B, Bq_3)$
$q_3 \sqcup \rightarrow q_4 \sqcup N$	(10)	$(q_3 \sqcup, q_4 \sqcup), (q_3  , q_4 \sqcup  )$

- Der akzeptierenden Rechnung

$$q_0 ab \underset{(1)}{\vdash} Aq_1 b \underset{(4)}{\vdash} q_2 AB \underset{(7)}{\vdash} Aq_0 B \underset{(8)}{\vdash} ABq_3 \sqcup \underset{10}{\vdash} ABq_4 \sqcup$$

von  $M(ab)$  entspricht dann das MPCP-Lösungswort

$$\langle | q_0 ab | Aq_1 b | q_2 AB | Aq_0 B | ABq_3 | ABq_4 \sqcup | Aq_4 \sqcup | q_4 \sqcup | q_4 | \rangle$$

$$\langle | q_0 ab | Aq_1 b | q_2 AB | Aq_0 B | ABq_3 | ABq_4 \sqcup | Aq_4 \sqcup | q_4 \sqcup | q_4 | \rangle$$

# Das Postsche Korrespondenzproblem

## Beweis von $A \leq \text{MPCP}_{\Sigma'}$

- Nun lässt sich leicht aus einer akzeptierenden Rechnung

$$K_0 = z_0 w \vdash K_1 \vdash \dots \vdash K_t = uev$$

mit  $e \in E$  und  $u, v \in \Gamma^*$  eine MPCP-Lösung mit einem Lösungswort der Form

$$\langle | K_0 | K_1 | \dots | K_t | K_{t+1} | \dots | K_{t+|K_t|-1} | \rangle$$

angeben, wobei  $K_{t+i}$  aus  $K_t$  durch Löschen von  $i$  Zeichen in der Nachbarschaft von  $e$  entsteht.

- Umgekehrt lässt sich aus jeder MPCP-Lösung auch eine akzeptierende Rechnung von  $M$  bei Eingabe  $w$  gewinnen, womit

$$w \in L(M) \Leftrightarrow f(w) \in \text{MPCP}_{\Sigma'}$$

gezeigt ist. □

# Das Schnittproblem für CFL ist unentscheidbar

## Das Schnittproblem für kontextfreie Grammatiken

Gegeben: Zwei kontextfreie Grammatiken  $G_1$  und  $G_2$ .

Gefragt: Ist  $L(G_1) \cap L(G_2) \neq \emptyset$ ?

## Satz

Das Schnittproblem für kontextfreie Grammatiken ist RE-vollständig.

## Beweis

- Es ist leicht zu sehen, dass das Problem semi-entscheidbar ist.
- Wir reduzieren eine PCP-Instanz  $I = \begin{pmatrix} x_1 \dots x_k \\ y_1 \dots y_k \end{pmatrix}$  auf ein Grammatikpaar  $(G_1, G_2)$ , so dass gilt:  $I \in \text{PCP} \Leftrightarrow L(G_1) \cap L(G_2) \neq \emptyset$ .
- Für  $i = 1, 2$  sei  $G_i = (\{S\}, \{a, b, 1, \dots, k\}, P_i, S)$  mit
$$P_1: S \rightarrow 1Sx_1, \dots, kSx_k, 1x_1, \dots, kx_k,$$
$$P_2: S \rightarrow 1Sy_1, \dots, kSy_k, 1y_1, \dots, ky_k.$$



# Das Schnittproblem für CFL ist unentscheidbar

## Beispiel

- Die PCP-Instanz

$$I = \begin{pmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{pmatrix} = \begin{pmatrix} a & aab & abbaa \\ aabba & ababb & aa \end{pmatrix}$$

wird auf das Grammatikpaar  $(G_1, G_2)$  mit folgenden Regeln reduziert:

$$P_1: S_1 \rightarrow 1Sa, 2Saab, 3Sabbaa, \\ 1a, 2aab, 3abbaa,$$

$$P_2: S_2 \rightarrow 1Saabba, 2Sababb, 3Saa, \\ 1aabba, 2ababb, 3aa.$$

- Der PCP-Lösung  $\alpha = (1, 3, 2, 3)$  entspricht dann das Wort

$$\begin{aligned} 3231x_1x_3x_2x_3 &= 3231aabbaa**a**ababbaa \\ &= 3231aabbaa**a**ababb**aa** = 3231y_1y_3y_2y_3 \end{aligned}$$

im Schnitt  $L(G_1) \cap L(G_2)$ .

# Das Schnittproblem für CFL ist unentscheidbar

## Reduktion von PCP auf das Schnittproblem für CFL

- Für  $i = 1, 2$  sei  $G_i = (\{S\}, \{a, b, 1, \dots, k\}, P_i, S)$  mit

$$P_1: S \rightarrow 1Sx_1, \dots, kSx_k, 1x_1, \dots, kx_k,$$

$$P_2: S \rightarrow 1Sy_1, \dots, kSy_k, 1y_1, \dots, ky_k.$$

- Dann gilt

$$L(G_1) = \{i_n \cdots i_1 x_{i_1} \cdots x_{i_n} \mid 1 \leq n, 1 \leq i_1, \dots, i_n \leq k\},$$

$$L(G_2) = \{i_n \cdots i_1 y_{i_1} \cdots y_{i_n} \mid 1 \leq n, 1 \leq i_1, \dots, i_n \leq k\}.$$

- Somit ist  $L(G_1) \cap L(G_2)$  die Sprache

$$\{i_n \cdots i_1 x_{i_1} \cdots x_{i_n} \mid 1 \leq n, x_{i_1} \cdots x_{i_n} = y_{i_1} \cdots y_{i_n}\}.$$

- Folglich ist  $\alpha = (i_1, \dots, i_n)$  genau dann eine Lösung für  $I$ , wenn  $i_n \cdots i_1 x_{i_1} \cdots x_{i_n} \in L(G_1) \cap L(G_2)$  ist.
- Also vermittelt  $f : I \mapsto (G_1, G_2)$  eine Reduktion von PCP auf das Schnittproblem für CFL.

# Das Schnitt- und das Inklusionsproblem für DCFL sind unentscheidbar

## Korollar

- 1 Das Schnittproblem für DPDAs ist RE-vollständig.
- 2 Das Inklusionsproblem für DPDAs ist co-RE-vollständig.

## Beweis.

- 1 Die kontextfreien Grammatiken  $G_1$  und  $G_2$  in obigem Beweis lassen sich leicht in äquivalente DPDAs  $M_1$  und  $M_2$  verwandeln (siehe Übungen).
- 2 Wir reduzieren das Komplement des Schnittproblems für DPDAs auf das Inklusionsproblem für DPDAs. Wegen

$$L_1 \cap L_2 = \emptyset \Leftrightarrow L_1 \subseteq \overline{L_2}.$$

berechnet die Funktion  $f : (M_1, M_2) \mapsto (M_1, \overline{M_2})$  die gewünschte Reduktion. □

# Weitere Unentscheidbarkeitsresultate für CFL

## Korollar

Für kontextfreie Grammatiken sind folgende Probleme unentscheidbar:

- 1 Ist  $L(G) = \Sigma^*$ ? (Ausschöpfungsproblem)
- 2 Ist  $L(G_1) = L(G_2)$ ? (Äquivalenzproblem)
- 3 Ist  $G$  mehrdeutig? (Mehrdeutigkeitsproblem)

### 1 Das Ausschöpfungsproblem für kf. Grammatiken ist co-RE-vollständig

Wir reduzieren das Komplement des Schnittproblems für DPDAs auf das Ausschöpfungsproblem für kontextfreie Grammatiken. Es gilt

$$L_1 \cap L_2 = \emptyset \Leftrightarrow \overline{L_1} \cup \overline{L_2} = \Sigma^*.$$

Daher vermittelt die Funktion  $f : (M_1, M_2) \mapsto G$ , wobei  $G$  eine kontextfreie Grammatik mit

$$L(G) = \overline{L(M_1)} \cup \overline{L(M_2)}$$

ist, die gewünschte Reduktion. □

# Weitere Unentscheidbarkeitsresultate für CFL

## Korollar

Für kontextfreie Grammatiken sind folgende Probleme unentscheidbar:

- 1 Ist  $L(G) = \Sigma^*$ ? (Ausschöpfungsproblem)
- 2 Ist  $L(G_1) = L(G_2)$ ? (Äquivalenzproblem)
- 3 Ist  $G$  mehrdeutig? (Mehrdeutigkeitsproblem)

## 2 Das Äquivalenzproblem für kontextfreie Grammatiken ist co-RE-vollständig

Wir reduzieren das Ausschöpfungsproblem für CFL auf das Äquivalenzproblem für CFL.

Dies leistet beispielsweise die Reduktionsfunktion

$$f : G \mapsto (G, G_{all}),$$

wobei  $G_{all}$  eine kontextfreie Grammatik mit  $L(G_{all}) = \Sigma^*$  ist. □

## Weitere Unentscheidbarkeitsresultate für CFL

### 3 Das Mehrdeutigkeitsproblem ist RE-vollständig

- Wir reduzieren PCP auf das Mehrdeutigkeitsproblem.
- Betrachte die Reduktionsfunktion  $f : \binom{x_1 \dots x_k}{y_1 \dots y_k} \mapsto G$  mit

$$G = (\{S, A, B\}, \{a, b, 1, \dots, k\}, P_1 \cup P_2 \cup \{S \rightarrow A, S \rightarrow B\}, S)$$

und den Regeln

$$P_1: A \rightarrow 1Ax_1, \dots, kAx_k, 1x_1, \dots, kx_k,$$

$$P_2: B \rightarrow 1By_1, \dots, kBy_k, 1y_1, \dots, ky_k.$$

- Da alle von  $A$  oder  $B$  ausgehenden Ableitungen eindeutig sind, ist  $G$  genau dann mehrdeutig, wenn es ein Wort  $w \in L(G)$  gibt mit

$$S \Rightarrow A \Rightarrow^* w \quad \text{und} \quad S \Rightarrow B \Rightarrow^* w.$$

- Wie wir im Beweis der Unentscheidbarkeit des Schnittproblems für CFL gesehen haben, ist dies genau dann der Fall, wenn die PCP-Instanz  $I = \binom{x_1 \dots x_k}{y_1 \dots y_k}$  eine PCP-Lösung hat. □

# Ein Unentscheidbarkeitsresultat für DCSL

## Das Leerheitsproblem für DLBAs

Gegeben: Ein DLBA  $M$ .

Gefragt: Ist  $L(M) = \emptyset$ ?

## Satz

Das Leerheitsproblem für DLBAs ist co-RE-vollständig.

## Beweis.

- Wir reduzieren das Ausschöpfungsproblem für CFL auf das Leerheitsproblem für DLBAs.
- Eine kontextfreie Grammatik  $G$  lässt sich wie folgt in einen DLBA  $M$  mit  $L(M) = \overline{L(G)}$  überführen (siehe Übungen):
  - Bestimme zunächst einen DLBA  $M$  mit  $L(M) = L(G)$ .
  - Konstruiere daraus einen DLBA  $\overline{M}$  mit  $L(\overline{M}) = \overline{L(M)}$ .
- Dann gilt  $L(G) = \Sigma^* \Leftrightarrow L(M) = \emptyset$ , d.h. die Funktion  $f : G \mapsto \overline{M}$  berechnet die gewünschte Reduktion.



# Entscheidbare Probleme

Dagegen ist es nicht schwer,

- für eine kontextfreie Grammatik  $G$  zu entscheiden, ob mindestens ein Wort in  $G$  ableitbar ist (Leerheitsproblem für CFL), und
- für eine kontextsensitive Grammatik  $G$  und ein Wort  $x$  zu entscheiden, ob  $x$  in  $G$  ableitbar ist (Wortproblem für CSL).

## Satz

- Das Leerheitsproblem für CFL ist entscheidbar.
- Das Wortproblem für CSL ist entscheidbar.

## Beweis.

Siehe Übungen.





# Überblick der gezeigten (Un-)Entscheidbarkeitsresultate

In folgender Tabelle fassen wir nochmals zusammen, wie schwierig die betrachteten Entscheidungsprobleme für die verschiedenen Stufen der Chomsky-Hierarchie sind.

	Wort- problem $x \in L?$	Leerheits- problem $L = \emptyset?$	Aus- schöpfung $L = \Sigma^*?$	Äquivalenz- problem $L_1 = L_2?$	Inklusions- problem $L_1 \subseteq L_2$	Schnitt- problem $L_1 \cap L_2 \neq \emptyset?$
REG	ja	ja	ja	ja	ja	ja
DCFL	ja	ja	ja	ja <sup>a</sup>	nein	nein
CFL	ja	ja	nein	nein	nein	nein
DCSL	ja	nein	nein	nein	nein	nein
CSL	ja	nein	nein	nein	nein	nein
RE	nein	nein	nein	nein	nein	nein

<sup>a</sup>Bewiesen in 1997 von Géraud Sénizergues (Univ. Bordeaux).

# Die arithmetische Hierarchie

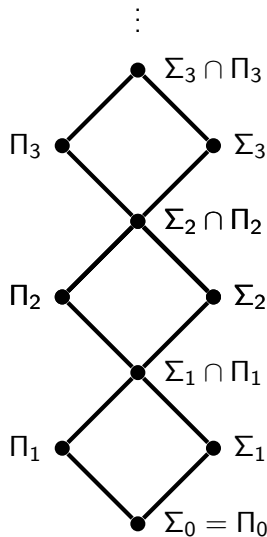
## Frage

Lässt sich der Grad der Unentscheidbarkeit von unentscheidbaren Problemen irgendwie messen?

## Definition

Sei  $A \subseteq \Sigma^*$  eine Sprache und  $\mathcal{K}$  eine Sprachklasse. Dann ist

- $\exists A = \{x \in \Sigma^* \mid \exists y \in \{0, 1\}^* : x\#y \in A\}$   
und
- $\exists \mathcal{K} = \mathcal{K} \cup \{\exists A \mid A \in \mathcal{K}\}$ .
- Weiter sei  $\Sigma_0 = \Pi_0 = \text{REC}$  und  $\Sigma_i = \exists \Pi_{i-1}$  sowie  $\Pi_i = \text{co-}\Sigma_i$  für  $i \geq 1$ .
- Die Sprachklassen  $\Sigma_i, \Pi_i, i \geq 0$ , bilden die Stufen der **arithmetischen Hierarchie**.



# Die arithmetische Hierarchie

## Proposition

- $RE = \Sigma_1$ .
- $\Sigma_i \cup \Pi_i \subseteq \Sigma_{i+1} \cap \Pi_{i+1}$  für  $i \geq 0$ .

## Beweis von $RE = \Sigma_1$

- Die Inklusion  $\exists REC \subseteq RE$  ist klar, da eine DTM im Fall  $B \in REC$  die Sprache  $A = \exists B$  akzeptiert, indem sie bei Eingabe  $x$  systematisch nach einem String  $y$  mit  $x\#y \in B$  sucht.
- Für den Beweis von  $RE \subseteq \exists REC$  sei  $A \subseteq \Sigma^*$  eine beliebige Sprache in  $RE$  und  $M$  sei eine DTM mit  $L(M) = A$ . Dann gilt  $A = \exists B$  für die Sprache

$$B = \{x\#y \mid x \in \Sigma^* \text{ und } y \text{ kodiert eine akz. Rechnung von } M(x)\}.$$

Da  $B$  offensichtlich entscheidbar ist, folgt  $A \in \exists REC$ . □

# Die arithmetische Hierarchie

## Beweis von $\Sigma_i \cup \Pi_i \subseteq \Sigma_{i+1} \cap \Pi_{i+1}$

- Die Inklusion  $\Pi_i \subseteq \Sigma_{i+1}$  ist klar, da  $\Pi_i \subseteq \exists \Pi_i = \Sigma_{i+1}$  ist.
- Wegen  $\mathcal{K} \subseteq \mathcal{K}' \Leftrightarrow \text{co-}\mathcal{K} \subseteq \text{co-}\mathcal{K}'$  und  $\Sigma_j = \text{co-}\Pi_j$  für  $j \geq 0$  folgt hieraus auch  $\Sigma_i \subseteq \Pi_{i+1}$ .
- Die Inklusionen  $\Sigma_{i-1} \subseteq \Sigma_i$  lassen sich induktiv wie folgt zeigen.
  - $i = 1$ :  $\Sigma_0 = \text{REC} \subseteq \text{RE} = \Sigma_1$ .
  - $i \rightsquigarrow i + 1$ : Nach IV gilt  $\Sigma_{i-1} \subseteq \Sigma_i$ . Folglich ist  $\Pi_{i-1} \subseteq \Pi_i$ , was wegen der Monotonie des  $\exists$ -Operators (d.h.  $\mathcal{K} \subseteq \mathcal{K}' \Rightarrow \exists \mathcal{K} \subseteq \exists \mathcal{K}'$ ) wiederum  $\Sigma_i = \exists \Pi_{i-1} \subseteq \exists \Pi_i = \Sigma_{i+1}$  impliziert.
- Durch Übergang zu den co-Klassen erhalten wir auch  $\Pi_{i-1} \subseteq \Pi_i$ . □

Mittels Diagonalisierung kann man zeigen, dass die Inklusionen  $\Sigma_{i-1} \subseteq \Sigma_i$  echt sind, und dass  $\Sigma_i \neq \Pi_i$  für alle  $i \geq 1$  gilt.

## Überblick der gezeigten (Un-)Entscheidbarkeitsresultate

Die betrachteten Entscheidungsprobleme für die verschiedenen Stufen der Chomsky-Hierarchie lassen sich nun wie folgt in die arithmetische Hierarchie einordnen.

	Wort- problem $x \in L?$	Leerheits- problem $L = \emptyset?$	Aus- schöpfung $L = \Sigma^*?$	Äquivalenz- problem $L_1 = L_2?$	Inklusions- problem $L_1 \subseteq L_2$	Schnitt- problem $L_1 \cap L_2 \neq \emptyset?$
REG	REC	REC	REC	REC	REC	REC
DCFL	REC	REC	REC	REC	co-RE	RE
CFL	REC	REC	co-RE	co-RE	co-RE	RE
DCSL	REC	co-RE	co-RE	co-RE	co-RE	RE
CSL	REC	co-RE	co-RE	co-RE	co-RE	RE
RE	RE	co-RE	$\Pi_2$	$\Pi_2$	$\Pi_2$	RE

Tatsächlich sind alle betrachteten Entscheidungsprobleme sogar vollständig für die angegebenen Sprachklassen.

# Zeitkomplexität von Turingmaschinen

Die Laufzeit einer NTM  $M$  bei Eingabe  $x$  ist die maximale Anzahl an Rechenschritten, die  $M(x)$  ausführt.

## Definition

- Die **Laufzeit** einer NTM  $M$  bei Eingabe  $x$  ist definiert als

$$\text{time}_M(x) = \max\{t \geq 0 \mid \exists K : K_x \vdash^t K\},$$

wobei  $\max \mathbb{N} = \infty$  ist.

- Sei  $t : \mathbb{N} \rightarrow \mathbb{N}$  eine monoton wachsende Funktion.
- Dann ist  $M$   **$t(n)$ -zeitbeschränkt**, falls für alle Eingaben  $x$  gilt:

$$\text{time}_M(x) \leq t(|x|).$$

Die Zeitschranke  $t(n)$  beschränkt also die Laufzeit bei allen Eingaben der Länge  $n$  (worst-case Komplexität).

# Zeitkomplexitätsklassen

Wir fassen alle Sprachen und Funktionen, die in einer vorgegebenen Zeitschranke  $t(n)$  entscheidbar bzw. berechenbar sind, in folgenden **Komplexitätsklassen** zusammen.

## Definition

- Die in deterministischer Zeit  $t(n)$  entscheidbaren Sprachen bilden die Sprachklasse

$$\text{DTIME}(t(n)) = \{L(M) \mid M \text{ ist eine } t(n)\text{-zeitbeschränkte DTM}\}.$$

- Die in nichtdeterministischer Zeit  $t(n)$  entscheidbaren Sprachen bilden die Sprachklasse

$$\text{NTIME}(t(n)) = \{L(M) \mid M \text{ ist eine } t(n)\text{-zeitbeschränkte NTM}\}.$$

- Die in deterministischer Zeit  $t(n)$  berechenbaren Funktionen bilden die Funktionenklasse

$$\text{FTIME}(t(n)) = \left\{ f \mid \begin{array}{l} \text{es gibt eine } t(n)\text{-zeitbeschränkte} \\ \text{DTM } M, \text{ die } f \text{ berechnet} \end{array} \right\}.$$

# Die wichtigsten Zeitkomplexitätsklassen

- Die wichtigsten deterministischen Zeitkomplexitätsklassen sind

$$\text{LINTIME} = \bigcup_{c \geq 1} \text{DTIME}(cn + c) \quad \text{„Linearzeit“}$$

$$\text{P} = \bigcup_{c \geq 1} \text{DTIME}(n^c + c) \quad \text{„Polynomialzeit“}$$

$$\text{E} = \bigcup_{c \geq 1} \text{DTIME}(2^{cn+c}) \quad \text{„Lineare Exponentialzeit“}$$

$$\text{EXP} = \bigcup_{c \geq 1} \text{DTIME}(2^{n^c+c}) \quad \text{„Exponentialzeit“}$$

- Die nichtdeterministischen Klassen **NLINTIME**, **NP**, **NE**, **NEXP** und die Funktionenklassen **FP**, **FE**, **FEXP** sind analog definiert.
- Für eine Klasse  $\mathcal{F}$  von Funktionen sei  $\text{DTIME}(\mathcal{F}) = \bigcup_{t \in \mathcal{F}} \text{DTIME}(t(n))$  (die Klassen  $\text{NTIME}(\mathcal{F})$  und  $\text{FTIME}(\mathcal{F})$  seien analog definiert).



# Asymptotische Laufzeit und Landau-Notation

## Definition

Seien  $f$  und  $g$  Funktionen von  $\mathbb{N}$  nach  $\mathbb{R}^+$ .

- Wir schreiben  $f(n) = O(g(n))$ , falls es Zahlen  $n_0$  und  $c$  gibt mit

$$\forall n \geq n_0 : f(n) \leq c \cdot g(n).$$

**Bedeutung:** „ $f$  wächst **nicht wesentlich schneller** als  $g$ .“

- Formal bezeichnet der Term  $O(g(n))$  die Klasse aller Funktionen  $f$ , die obige Bedingung erfüllen.
- Die Gleichung  $f(n) = O(g(n))$  drückt also in Wahrheit eine **Element-Beziehung**  $f \in O(g(n))$  aus.
- $O$ -Terme können auch auf der linken Seite vorkommen.
- In diesem Fall wird eine **Inklusionsbeziehung** ausgedrückt.
- So steht  $n^2 + O(n) = O(n^2)$  für die Aussage

$$\{n^2 + f \mid f \in O(n)\} \subseteq O(n^2).$$

# Asymptotische Laufzeit und Landau-Notation

## Beispiel

- $7 \log(n) + n^3 = O(n^3)$  ist **richtig**.
- $7 \log(n)n^3 = O(n^3)$  ist **falsch**.
- $2^{n+O(1)} = O(2^n)$  ist **richtig**.
- $2^{O(n)} = O(2^n)$  ist **falsch** (siehe Übungen).

Mit der  $O$ -Notation lassen sich die wichtigsten deterministischen Zeitkomplexitätsklassen wie folgt charakterisieren:

**L**INTIME = DTIME( $O(n)$ ) „Linearzeit“

**P** = DTIME( $n^{O(1)}$ ) „Polynomialzeit“

**E** = DTIME( $2^{O(n)}$ ) „Lineare Exponentialzeit“

**EXP** = DTIME( $2^{n^{O(1)}}$ ) „Exponentialzeit“

# Platzkomplexität von Turingmaschinen

- Als nächstes definieren wir den Platzverbrauch von NTMs.
- Intuitiv ist dies die Anzahl aller während einer Rechnung benutzten Bandfelder.
- Wollen wir auch sublinearen Platz sinnvoll definieren, so dürfen wir hierbei das Eingabeband offensichtlich nicht berücksichtigen.
- Um sicherzustellen, dass eine NTM  $M$  das Eingabeband nicht als Speicher benutzt, verlangen wir, dass
  - $M$  die Felder auf dem Eingabeband nicht verändert und
  - sich nicht mehr als ein Feld von der Eingabe entfernt.

## Definition

Eine NTM  $M$  heißt **offline-NTM** (oder NTM mit **Eingabeband**), falls für jede von  $M$  bei Eingabe  $x$  erreichbare Konfiguration

$$K = (q, u_1, a_1, v_1, \dots, u_k, a_k, v_k)$$

gilt, dass  $u_1 a_1 v_1$  ein Teilwort von  $\sqcup x \sqcup$  ist.

# Platzkomplexität von Turingmaschinen

## Definition

- Der **Platzverbrauch** einer offline-NTM  $M$  bei Eingabe  $x$  ist definiert als

$$space_M(x) = \max \left\{ s \geq 1 \left| \begin{array}{l} \exists K = (q, u_1, a_1, v_1, \dots, u_k, a_k, v_k) \\ \text{mit } K_x \vdash^* K \text{ und } s = \sum_{i=2}^k |u_i a_i v_i| \end{array} \right. \right\}.$$

- Sei  $s : \mathbb{N} \rightarrow \mathbb{N}$  eine monoton wachsende Funktion.
- $M$  heißt  **$s(n)$ -platzbeschränkt**, falls für alle Eingaben  $x$  gilt:

$$space_M(x) \leq s(|x|).$$

# Platzkomplexitätsklassen

Wir fassen alle Sprachen, die in einer vorgegebenen Platzschranke  $s(n)$  entscheidbar sind, in folgenden **Platzkomplexitätsklassen** zusammen.

## Definition

- Die auf deterministischem Platz  $s(n)$  entscheidbaren Sprachen bilden die Klasse

$$\mathbf{DSPACE}(s(n)) = \{L(M) \mid M \text{ ist eine } s(n)\text{-platzb. offline-DTM}\}.$$

- Die auf nichtdeterministischem Platz  $s(n)$  entscheidbaren Sprachen bilden die Klasse

$$\mathbf{NSPACE}(s(n)) = \{L(M) \mid M \text{ ist eine } s(n)\text{-platzb. offline-NTM}\}.$$

# Die wichtigsten Platzkomplexitätsklassen

- Die wichtigsten deterministischen Platzkomplexitätsklassen sind

$$L = \text{DSPACE}(O(\log n)) \quad \text{„Logarithmischer Platz“}$$

$$L\text{INSPACE} = \text{DSPACE}(O(n)) \quad \text{„Linearer Platz“}$$

$$PSPACE = \text{DSPACE}(n^{O(1)}) \quad \text{„Polynomieller Platz“}$$

- Die nichtdeterministischen Klassen **NL**, **NLINSPACE** und **NPSPACE** sind analog definiert.
- Der Satz von Savitch besagt, dass

$$NPSPACE(s(n)) \subseteq \text{DSPACE}(s^2(n))$$

ist (dies wird in der VL Komplexitätstheorie gezeigt).

Daher fallen die Klassen NPSPACE und PSPACE zusammen.

# Die wichtigsten Zeit- und Platzkomplexitätsklassen

Die wichtigsten Zeitkomplexitätsklassen sind

**LINTIME** =  $\text{DTIME}(O(n))$  „Linearzeit“

**P** =  $\text{DTIME}(n^{O(1)})$  „Polynomialzeit“

**NP** =  $\text{NTIME}(n^{O(1)})$  „Nichtdet. Polynomialzeit“

**E** =  $\text{DTIME}(2^{O(n)})$  „Lineare Exponentialzeit“

**EXP** =  $\text{DTIME}(2^{n^{O(1)}})$  „Exponentialzeit“

Die wichtigsten Platzkomplexitätsklassen sind

**L** =  $\text{DSPACE}(O(\log n))$  „Logarithmischer Platz“

**NL** =  $\text{NSPACE}(O(\log n))$  „Nichtdet. logarithmischer Platz“

**LINSPACE** =  $\text{DSPACE}(O(n))$  „Linearer Platz“

**PSPACE** =  $\text{DSPACE}(n^{O(1)})$  „Polynomieller Platz“

**EXPSPACE** =  $\text{DSPACE}(2^{n^{O(1)}})$  „Exponentialer Platz“

# Elementare Beziehungen zwischen Komplexitätsklassen

## Frage

Welche elementaren Beziehungen gelten zwischen den verschiedenen Zeit- und Platzklassen?

## Satz

- Für jede Funktion  $s(n) \geq \log n$  gilt

$$\text{DSPACE}(s) \subseteq \text{NSPACE}(s) \subseteq \text{DTIME}(2^{O(s)}).$$

- Für jede Funktion  $t(n) \geq n + 2$  gilt

$$\text{DTIME}(t) \subseteq \text{NTIME}(t) \subseteq \text{DSPACE}(t).$$

## Korollar

Es gilt  $L \subseteq NL \subseteq P \subseteq NP \subseteq PSPACE \subseteq EXP \subseteq EXPSPACE$ .



# Komplexitätsschranken für die Stufen der Chomsky-Hierarchie

$$\text{REG} = \text{DSPACE}(O(1)) = \text{NSPACE}(O(1)) \subsetneq \text{L},$$

$$\text{DCFL} \subsetneq \text{LINTIME},$$

$$\text{CFL} \subsetneq \text{NLINTIME} \cap \text{DTIME}(O(n^3)) \subsetneq \text{P},$$

$$\text{DCSL} = \text{LINSPACE} \subseteq \text{CSL},$$

$$\text{CSL} = \text{NLINSPACE} \subseteq \text{PSPACE} \cap \text{E},$$

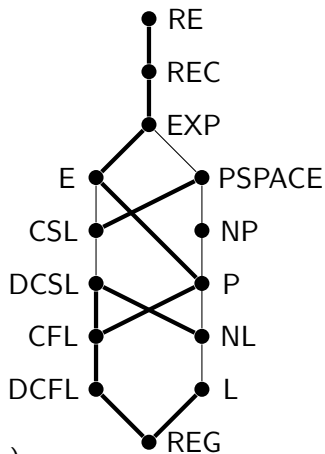
$$\text{REC} = \bigcup_f \text{DSPACE}(f(n))$$

$$= \bigcup_f \text{NSPACE}(f(n))$$

$$= \bigcup_f \text{DTIME}(f(n))$$

$$= \bigcup_f \text{NTIME}(f(n)),$$

wobei  $f$  alle (oder äquivalent: alle berechenbaren)  
Funktionen  $f : \mathbb{N} \rightarrow \mathbb{N}$  durchläuft.



# Das P-NP-Problem

- Wie wir gesehen haben, sind NTMs nicht mächtiger als DTMs, d.h. jede NTM kann von einer DTM simuliert werden.
- Die Frage, wieviel Zeit eine DTM zur Simulation einer NTM benötigt, ist eines der wichtigsten offenen Probleme der Informatik.
- Wegen  $\text{NTIME}(t) \subseteq \text{DSpace}(t) \subseteq \text{NSpace}(t) \subseteq \text{DTIME}(2^{O(t)})$  erhöht sich die Laufzeit im schlimmsten Fall exponentiell.
- Insbesondere die Klasse NP enthält viele für die Praxis überaus wichtige Probleme, für die kein Polynomialzeitalgorithmus bekannt ist.
- Da jedoch nur Probleme in P als effizient lösbar angesehen werden, hat das so genannte **P-NP-Problem**, also die Frage, ob alle NP-Probleme effizient lösbar sind, eine immense praktische Bedeutung.
- Wie bereits erwähnt, ist diese Frage für die Platzkomplexität bereits gelöst (Satz von Savitch, 1970).

# Die Polynomialzeitreduktion

## Definition

- Eine Sprache  $A \subseteq \Sigma^*$  ist auf  $B \subseteq \Gamma^*$  **in Polynomialzeit reduzierbar** ( $A \leq^P B$ ), falls eine Funktion  $f : \Sigma^* \rightarrow \Gamma^*$  in FP existiert mit

$$\forall x \in \Sigma^* : x \in A \Leftrightarrow f(x) \in B.$$

- Eine Sprache  $A$  heißt  **$\leq^P$ -hart** für eine Sprachklasse  $\mathcal{C}$  (kurz:  **$\mathcal{C}$ -hart** oder  **$\mathcal{C}$ -schwer**), falls gilt:

$$\forall L \in \mathcal{C} : L \leq^P A.$$

- Eine  $\mathcal{C}$ -harte Sprache  $A$ , die zu  $\mathcal{C}$  gehört, heißt  **$\mathcal{C}$ -vollständig**.
- **NPC** bezeichnet die Klasse aller NP-vollständigen Sprachen.

## Lemma

- Aus  $A \leq^P B$  folgt  $A \leq B$ .
- Die Reduktionsrelation  $\leq^P$  ist reflexiv und transitiv (s. Übungen).

# Die Polynomialzeitreduktion

## Satz

Die Klassen P und NP sind unter  $\leq^P$  abgeschlossen.

## Beweis

- Sei  $B \in P$  und gelte  $A \leq^P B$  mittels einer Funktion  $f \in FP$ .
- Seien  $M$  und  $T$  DTMs mit  $L(M) = B$  und  $T(x) = f(x)$ .
- Weiter seien  $p$  und  $q$  polynomielle Zeitschranken für  $M$  und  $T$ .
- Betrachte die DTM  $M'$ , die bei Eingabe  $x$  zuerst  $T$  simuliert, um  $f(x)$  zu berechnen, und danach  $M$  bei Eingabe  $f(x)$  simuliert.
- Dann gilt

$$x \in A \Leftrightarrow f(x) \in B \Leftrightarrow f(x) \in L(M) \Leftrightarrow x \in L(M').$$

- Also ist  $L(M') = A$  und wegen

$$\text{time}_{M'}(x) \leq \text{time}_T(x) + \text{time}_M(f(x)) \leq q(|x|) + p(q(|x|))$$

ist  $M'$  polynomiell zeitbeschränkt und somit  $A$  in P. □

# NP-Vollständigkeit

## Satz

- 1  $A \leq^P B$  und  $A$  ist NP-hart  $\Rightarrow B$  ist NP-hart.
- 2  $A \leq^P B$ ,  $A$  ist NP-hart und  $B \in \text{NP}$   $\Rightarrow B \in \text{NPC}$ .
- 3  $\text{NPC} \cap \text{P} \neq \emptyset \Rightarrow \text{P} = \text{NP}$ .

## Beweis

- 1 Da  $A$  NP-hart ist, ist jede NP-Sprache  $L$  auf  $A$  reduzierbar. Da zudem  $A \leq^P B$  gilt und  $\leq^P$  transitiv ist, folgt  $L \leq^P B$ .
- 2 Klar, da  $B$  mit (1) NP-hart und nach Voraussetzung in NP ist.
- 3 Sei  $B$  eine NP-vollständige Sprache in P. Dann ist jede NP-Sprache  $A$  auf  $B$  reduzierbar und da P unter  $\leq^P$  abgeschlossen ist, folgt  $A \in \text{P}$ .  $\square$

# NP-Vollständigkeit

## Satz

Folgende Sprache ist NP-vollständig:

$$L = \left\{ w\#x\#0^m \mid \begin{array}{l} w, x \in \{0, 1\}^* \text{ und } M_w \text{ ist eine NTM,} \\ \text{die } x \text{ in } \leq m \text{ Schritten akzeptiert} \end{array} \right\}$$

## Beweis

- Zunächst ist klar, dass  $L \in \text{NP}$  mittels folgender NTM  $M$  ist:  
 $M$  simuliert bei Eingabe  $w\#x\#0^m$  die NTM  $M_w$  bei Eingabe  $x$  für höchstens  $m$  Schritte. Falls  $M_w$  in dieser Zeit akzeptiert, akzeptiert  $M$  ebenfalls, andernfalls verwirft  $M$ .
- Sei nun  $A$  eine beliebige NP-Sprache. Dann ist  $A$  in Polynomialzeit auf eine Sprache  $B \subseteq \{0, 1\}^*$  in NP reduzierbar (siehe Übungen).
- Sei  $M_w$  eine durch ein Polynom  $p$  zeitbeschränkte NTM für  $B$ .
- Dann reduziert folgende FP-Funktion  $f$  die Sprache  $B$  auf  $L$ :

$$f : x \mapsto w\#x\#0^{p(|x|)}.$$



# Aussagenlogische Formeln

- Die Menge der **booleschen** (oder **aussagenlogischen**) **Formeln** über den Variablen  $x_1, \dots, x_n$ ,  $n \geq 0$ , ist induktiv wie folgt definiert:
  - Die Konstanten 0 und 1 sind boolesche Formeln.
  - Jede Variable  $x_i$  ist eine boolesche Formel.
  - Mit  $G$  und  $H$  sind auch die **Negation**  $\neg G$ , die **Konjunktion**  $(G \wedge H)$  und die **Disjunktion**  $(G \vee H)$  von  $G$  und  $H$  Formeln.
- Eine **Belegung** von  $x_1, \dots, x_n$  ist ein Wort  $a = a_1 \dots a_n \in \{0, 1\}^n$ .
- Der **Wert**  $F(a)$  von  $F$  unter  $a$  ist induktiv wie folgt definiert:

---

$F$	0	1	$x_i$	$\neg G$	$(G \wedge H)$	$(G \vee H)$
$F(a)$	0	1	$a_i$	$1 - G(a)$	$G(a)H(a)$	$G(a) + H(a) - G(a)H(a)$

---

- Durch die Formel  $F$  wird also eine  **$n$ -stellige boolesche Funktion**  $F : \{0, 1\}^n \rightarrow \{0, 1\}$  definiert, die wir ebenfalls mit  $F$  bezeichnen.
- $F$  heißt **erfüllbar**, falls es eine Belegung  $a$  mit  $F(a) = 1$  gibt.

# Aussagenlogische Formeln

## Notation

Wir benutzen die **Implikation**  $G \rightarrow H$  als Abkürzung für die Formel  $\neg G \vee H$ .

## Beispiel (Erfüllbarkeitstest mittels Wahrheitstabelle)

Die Formel  $((\neg x_1 \vee \neg x_2) \rightarrow (x_2 \wedge x_3))$  ist erfüllbar:

$a$	$(\neg x_1 \vee \neg x_2)$	$(x_2 \wedge x_3)$	$((\neg x_1 \vee \neg x_2) \rightarrow (x_2 \wedge x_3))$
000	1	0	0
001	1	0	0
010	1	0	0
011	1	1	1
100	1	0	0
101	1	0	0
110	0	0	1
111	0	1	1



# Aussagenlogische Formeln

## Präzedenzregeln zur Klammerersparnis

- Der Junktor  $\wedge$  bindet stärker als der Junktor  $\vee$  und dieser wiederum stärker als der Junktor  $\rightarrow$ .
- Formeln der Form  $(x_1 \circ (x_2 \circ (x_3 \circ \dots \circ x_n) \dots))$ ,  $\circ \in \{\wedge, \vee, \rightarrow\}$  kürzen wir durch  $(x_1 \circ \dots \circ x_n)$  ab.

## Beispiel (Formel für die mehrstellige Entweder-Oder Funktion)

- Folgende Formel nimmt unter einer Belegung  $a = a_1 \cdots a_n$  genau dann den Wert 1 an, wenn  $\sum_{i=1}^n a_i = 1$  ist:

$$G(x_1, \dots, x_n) = (x_1 \vee \dots \vee x_n) \wedge \bigwedge_{1 \leq i < j \leq n} \neg(x_i \wedge x_j)$$

- D.h. es gilt genau dann  $G(a) = 1$ , wenn genau eine Variable  $x_i$  mit dem Wert  $a_i = 1$  belegt ist.
- Diese Formel wird im Beweis des nächsten Satzes benötigt. ◀

# Das aussagenlogische Erfüllbarkeitsproblem

Erfüllbarkeitsproblem für boolesche Formeln (*satisfiability*, SAT):

Gegeben: Eine boolesche Formel  $F$ .

Gefragt: Ist  $F$  erfüllbar?

Satz (Cook, Karp, Levin)

SAT ist NP-vollständig.

Lässt sich also ein effizienter Algorithmus für SAT finden, so kann daraus leicht ein effizienter Algorithmus für jedes NP-Problem abgeleitet werden.

Korollar

$\text{SAT} \in \text{P} \Leftrightarrow \text{P} = \text{NP}$ .

# SAT ist NP-vollständig

## Beweis von $\text{SAT} \in \text{NP}$

Es ist leicht zu sehen, dass  $\text{SAT} \in \text{NP}$  ist:

Eine NTM kann bei Eingabe einer booleschen Formel  $F$  zunächst eine Belegung  $a$  nichtdeterministisch raten und dann in Polynomialzeit testen, ob  $F(a) = 1$  ist (*guess and verify* Strategie). □

## Beweis von SAT ist NP-hart

- Sei  $L$  eine beliebige NP-Sprache und sei  $M = (Z, \Sigma, \Gamma, \delta, q_0)$  eine durch ein Polynom  $p$  zeitbeschränkte  $k$ -NTM mit  $L(M) = L$ .
- Da sich eine  $t(n)$ -zeitbeschränkte  $k$ -NTM in Zeit  $t^2(n)$  durch eine 1-NTM simulieren lässt, können wir  $k = 1$  annehmen.
- Unsere Aufgabe besteht nun darin, in Polynomialzeit zu einer gegebenen Eingabe  $w = w_1 \cdots w_n$  eine Formel  $F_w$  zu konstruieren, die genau dann erfüllbar ist, wenn  $w \in L$  ist.
- O.B.d.A. sei  $Z = \{q_0, \dots, q_m\}$ ,  $E = \{q_m\}$  und  $\Gamma = \{a_1, \dots, a_l\}$ .
- Zudem sei  $\delta(q_m, a) := \{(q_m, a, N)\}$  für alle  $a \in \Gamma$ .

# Beweis von SAT ist NP-hart

## Idee:

Konstruiere  $F_w$  so, dass  $F_w$  unter einer Belegung  $a$  genau dann wahr wird, wenn  $a$  eine akzeptierende Rechnung von  $M(w)$  beschreibt.

- Wir bilden  $F_w$  über den Variablen

$$x_{t,q}, \quad \text{für } 0 \leq t \leq p(n), q \in Z,$$

$$y_{t,i}, \quad \text{für } 0 \leq t \leq p(n), -p(n) \leq i \leq p(n),$$

$$z_{t,i,a}, \quad \text{für } 0 \leq t \leq p(n), -p(n) \leq i \leq p(n), a \in \Gamma.$$

- Diese Variablen stehen für folgende Aussagen:

$x_{t,q}$ : zum Zeitpunkt  $t$  befindet sich  $M$  im Zustand  $q$ ,

$y_{t,i}$ : zur Zeit  $t$  besucht  $M$  das Feld mit der Nummer  $i$ ,

$z_{t,i,a}$ : zur Zeit  $t$  steht das Zeichen  $a$  auf dem  $i$ -ten Feld.

- Konkret sei  $F_w = R \wedge S_w \wedge \dot{U}_1 \wedge \dot{U}_2 \wedge E$ .

## Beweis von SAT ist NP-hart

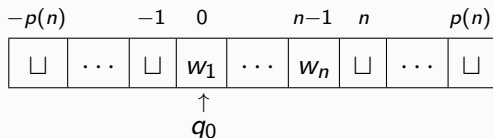
- Konkret sei  $F_w = R \wedge S_w \wedge \ddot{U}_1 \wedge \ddot{U}_2 \wedge E$ .
- Dabei stellt die Formel  $R = \bigwedge_{t=0}^{p(n)} R_t$  (Randbedingungen) sicher, dass wir jeder erfüllenden Belegung eindeutig eine Folge von Konfigurationen  $K_0, \dots, K_{p(n)}$  zuordnen können:

$$R_t = G(x_{t,q_0}, \dots, x_{t,q_m}) \wedge G(y_{t,-p(n)}, \dots, y_{t,p(n)}) \\ \wedge \bigwedge_{i=-p(n)}^{p(n)} G(z_{t,i,a_1}, \dots, z_{t,i,a_l}).$$

- Die Teilformel  $R_t$  sorgt also dafür, dass zum Zeitpunkt  $t$ 
  - genau ein Zustand  $q \in \{q_0, \dots, q_m\}$  eingenommen wird,
  - genau ein Bandfeld  $i \in \{-p(n), \dots, p(n)\}$  besucht wird und
  - auf jedem Feld  $i$  genau ein Zeichen  $a \in \Gamma$  steht.

## Beweis von SAT ist NP-hart

- Die Formel  $S_w$  (wie Startbedingung) stellt sicher, dass zum Zeitpunkt 0 tatsächlich die Startkonfiguration vorliegt:



$$S_w = x_{0,q_0} \wedge y_{0,0} \wedge \bigwedge_{i=-p(n)}^{-1} z_{0,i,\sqcup} \wedge \bigwedge_{i=0}^{n-1} z_{0,i,w_{i+1}} \wedge \bigwedge_{i=n}^{p(n)} z_{0,i,\sqcup}$$

- Die Formel  $\ddot{U}_1$  sorgt dafür, dass der Inhalt von nicht besuchten Feldern beim Übergang von  $K_t$  zu  $K_{t+1}$  unverändert bleibt:

$$\ddot{U}_1 = \bigwedge_{t=0}^{p(n)-1} \bigwedge_{i=-p(n)}^{p(n)} \bigwedge_{a \in \Gamma} (\neg y_{t,i} \wedge z_{t,i,a} \rightarrow z_{t+1,i,a})$$

## Beweis von SAT ist NP-hart

- $\ddot{U}_2$  achtet darauf, dass sich bei jedem Übergang der Zustand, die Kopfposition und das gerade gelesene Zeichen gemäß einer Anweisung in  $\delta$  verändern:

$$\ddot{U}_2 = \bigwedge_{t=0}^{p(n)-1} \bigwedge_{i=-p(n)}^{p(n)} \bigwedge_{a \in \Gamma} \bigwedge_{p \in Z} (x_{t,p} \wedge y_{t,i} \wedge z_{t,i,a} \rightarrow$$

$$\bigvee_{(q,b,D) \in \delta(p,a)} x_{t+1,q} \wedge y_{t+1,i+D} \wedge z_{t+1,i,b}),$$

wobei

$$i + D = \begin{cases} i - 1, & D = L \\ i, & D = N \\ i + 1, & D = R \end{cases}$$

- Schließlich überprüft  $E$ , ob  $M$  zur Zeit  $p(n)$  den Endzustand  $q_m$  erreicht hat:

$$E = x_{p(n),q_m}$$

## Beweis von SAT ist NP-hart

- Da der Aufbau der Formel  $f(w) = F_w$  einem einfachen Bildungsgesetz folgt und ihre Länge polynomiell in  $n$  ist, folgt  $f \in \text{FP}$ .
- Es ist klar, dass  $F_w$  im Fall  $w \in L(M)$  erfüllbar ist, indem wir die Variablen von  $F_w$  gemäß einer akz. Rechnung von  $M(w)$  belegen.
- Umgekehrt führt eine Belegung  $a$  mit  $F_w(a) = 1$  wegen  $R(a) = 1$  eindeutig auf eine Konfigurationenfolge  $K_0, \dots, K_{p(n)}$ , so dass gilt:
  - $K_0$  ist Startkonfiguration von  $M(w)$  (wegen  $S_w(a) = 1$ ),
  - $K_i \vdash K_{i+1}$  für  $i = 0, \dots, p(n) - 1$  (wegen  $\ddot{U}_1(a) = \ddot{U}_2(a) = 1$ ),
  - $M$  nimmt spätestens bei Erreichen der Konfiguration  $K_{p(n)}$  den Endzustand  $q_m$  an (wegen  $E(a) = 1$ ).
- Also gilt für alle  $w \in \Sigma^*$  die Äquivalenz

$$w \in L(M) \Leftrightarrow F_w \in \text{SAT},$$

d.h. die FP-Funktion  $f : w \mapsto F_w$  reduziert  $L(M)$  auf SAT. □



# Boolesche Schaltkreise

Als nächstes betrachten wir das Erfüllbarkeitsproblem für boolesche Schaltkreise.

## Definition

- Ein **boolescher Schaltkreis** über den Variablen  $x_1, \dots, x_n$  ist eine Folge  $s = (g_1, \dots, g_m)$  von **Gattern**

$$g_l \in \{0, 1, x_1, \dots, x_n, (\neg, j), (\wedge, j, k), (\vee, j, k)\} \text{ mit } 1 \leq j, k < l.$$

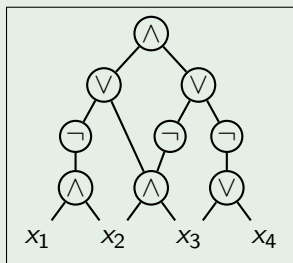
- Die am Gatter  $g_l$  berechnete  $n$ -stellige boolesche Funktion ist induktiv wie folgt definiert:

$g_l$	0	1	$x_i$	$(\neg, j)$	$(\wedge, j, k)$	$(\vee, j, k)$
$g_l(a)$	0	1	$a_i$	$1 - g_j(a)$	$g_j(a)g_k(a)$	$g_j(a) + g_k(a) - g_j(a)g_k(a)$

- $s$  berechnet die boolesche Funktion  $s(a) = g_m(a)$ .
- $s$  heißt **erfüllbar**, wenn eine Eingabe  $a \in \{0, 1\}^n$  mit  $s(a) = 1$  ex.

# Boolesche Schaltkreise

## Beispiel



Graphische Darstellung

des Schaltkreises  $c =$

$(x_1, x_2, x_3, x_4, (\wedge, 1, 2),$   
 $(\wedge, 2, 3), (\vee, 3, 4), (\neg, 5),$   
 $(\neg, 6), (\neg, 7), (\vee, 6, 8),$   
 $(\vee, 9, 10), (\wedge, 11, 12)).$

## Bemerkung

- Die Anzahl der Eingänge eines Gatters  $g$  wird als **Fanin** von  $g$  bezeichnet,
- die Anzahl der Ausgänge von  $g$  (also die Anzahl der Gatter, die  $g$  als Eingabe benutzen) als **Fanout**.
- Boolesche Formeln entsprechen also den booleschen Schaltkreisen mit (maximalem) Fanout 1 und umgekehrt.
- Eine boolesche Formel  $F$  kann somit leicht in einen äquivalenten Schaltkreis  $s$  mit  $s(a) = F(a)$  für alle Belegungen  $a$  transformiert werden.

# Das Erfüllbarkeitsproblem für boolesche Schaltkreise

Erfüllbarkeitsproblem für boolesche Schaltkreise (CIRSAT):

Gegeben: Ein boolescher Schaltkreis  $s$ .

Gefragt: Ist  $s$  erfüllbar?

## Satz

CIRSAT ist NP-vollständig.

## Beweis

Klar, da  $SAT \leq^P CIRSAT$  und  $CIRSAT \in NP$  gilt. □

## Bemerkung

- Da SAT NP-vollständig ist, ist CIRSAT auf SAT reduzierbar.
- Dies bedeutet jedoch nicht, dass sich jeder Schaltkreis in Polynomialzeit in eine logisch äquivalente Formel überführen lässt, sondern nur in eine erfüllbarkeitsäquivalente Formel.
- CIRSAT ist sogar auf eine spezielle SAT-Variante reduzierbar.

# Formeln in konjunktiver Normalform (KNF)

## Definition

- Ein **Literal** ist eine Variable  $x_i$  oder eine negierte Variable  $\neg x_i$ , die wir auch kurz mit  $\bar{x}_i$  bezeichnen.
- Eine **Klausel** ist eine Disjunktion  $C = \bigvee_{j=1}^k l_j$  von Literalen.
- Eine boolesche Formel  $F$  ist in **konjunktiver Normalform** (kurz **KNF**), falls  $F$  eine Konjunktion von Klauseln  $C_1, \dots, C_m$  ist,

$$F = \bigwedge_{j=1}^m C_j.$$

- Enthält jede Klausel höchstens  $k$  Literale, so heißt  $F$  in  **$k$ -KNF**.

## Notation

Klauseln werden oft als Menge  $C = \{l_1, \dots, l_k\}$  ihrer Literale und KNF-Formeln als Menge  $F = \{C_1, \dots, C_m\}$  ihrer Klauseln dargestellt.

# Erfüllbarkeitsprobleme für Formeln in KNF

## Erfüllbarkeitsproblem für $k$ -KNF Formeln ( $k$ -SAT):

Gegeben: Eine boolesche Formel  $F$  in  $k$ -KNF.

Gefragt: Ist  $F$  erfüllbar?

## Not-All-Equal-SAT ( $\text{NAESAT}$ ):

Gegeben: Eine Formel  $F$  in 3-KNF.

Gefragt: Hat  $F$  eine (erfüllende) Belegung, unter der in keiner Klausel alle Literale denselben Wahrheitswert haben?

## Beispiel

- Die 3-KNF Formel  $F = (x_1 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee x_3) \wedge (x_2 \vee \bar{x}_3 \vee x_4)$  ist alternativ durch folgende Klauselmenge darstellbar:

$$F = \{\{x_1, \bar{x}_2\}, \{\bar{x}_1, x_3\}, \{x_2, \bar{x}_3, x_4\}\}$$

- Offenbar ist  $F(1111) = 1$ , d.h.  $F \in 3\text{-SAT}$ . Zudem gilt  $F \in \text{NAESAT}$ .

# Komplexität von Erfüllbarkeitsproblemen

## Satz

- 1-SAT und 2-SAT sind in P entscheidbar (siehe Übungen).
- 3-SAT und NAE-SAT sind NP-vollständig.

## 3-SAT ist NP-vollständig

### Reduktion von CIRSAT auf 3-SAT

- Wir transformieren einen Schaltkreis  $s = (g_1, \dots, g_m)$  mit  $n$  Eingängen in eine 3-KNF Formel  $F_s$  über den Variablen  $x_1, \dots, x_n, y_1, \dots, y_m$ , die die Klausel  $\{y_m\}$  und für jedes Gatter  $g_i$  die folgenden Klauseln enthält:

Gatter $g_i$	zugeh. Klauseln in $F_s$	Semantik
0	$\{\bar{y}_i\}$	$y_i = 0$
1	$\{y_i\}$	$y_i = 1$
$x_j$	$\{\bar{y}_i, x_j\}, \{\bar{x}_j, y_i\}$	$y_i \leftrightarrow x_j$
$(\neg, j)$	$\{\bar{y}_i, \bar{y}_j\}, \{y_j, y_i\}$	$y_i \leftrightarrow \bar{y}_j$
$(\wedge, j, k)$	$\{\bar{y}_i, y_j\}, \{\bar{y}_i, y_k\}, \{\bar{y}_j, \bar{y}_k, y_i\}$	$y_i \leftrightarrow y_j \wedge y_k$
$(\vee, j, k)$	$\{\bar{y}_j, y_i\}, \{\bar{y}_k, y_i\}, \{\bar{y}_i, y_j, y_k\}$	$y_i \leftrightarrow y_j \vee y_k$

## 3-SAT ist NP-vollständig

### Reduktion von CIRSAT auf 3-SAT

- Nun ist leicht zu sehen, dass für alle  $a \in \{0, 1\}^n$  folgende Äquivalenz gilt:

$$s(a) = 1 \Leftrightarrow \exists b \in \{0, 1\}^m : F_s(ab) = 1.$$

- Ist nämlich  $a \in \{0, 1\}^n$  eine Eingabe mit  $s(a) = 1$ . Dann erhalten wir mit

$$b_i = g_i(a) \text{ für } i = 1, \dots, m$$

eine erfüllende Belegung  $ab_1 \cdots b_m$  für  $F_s$ .

- Ist umgekehrt  $ab_1 \cdots b_m$  eine erfüllende Belegung für  $F_s$ , so muss
  - $b_m = 1$  sein, da  $\{y_m\}$  eine Klausel in  $F_s$  ist, und
  - durch Induktion über  $i = 1, \dots, m$  folgt

$$g_i(a) = b_i,$$

d.h. insbesondere folgt  $s(a) = g_m(a) = b_m = 1$ .



## 3-SAT ist NP-vollständig

### Reduktion von CIRSAT auf 3-SAT

- Nun ist leicht zu sehen, dass für alle  $a \in \{0, 1\}^n$  folgende Äquivalenz gilt:

$$s(a) = 1 \Leftrightarrow \exists b \in \{0, 1\}^m : F_s(ab) = 1.$$

- Dies bedeutet, dass der Schaltkreis  $s$  und die 3-KNF-Formel  $F_s$  erfüllbarkeitsäquivalent sind, d.h.

$$s \in \text{CIRSAT} \Leftrightarrow F_s \in \text{3-SAT}.$$

- Zudem ist leicht zu sehen, dass die Reduktionsfunktion  $s \mapsto F_s$  in FP berechenbar ist, womit  $\text{CIRSAT} \leq^p \text{3-SAT}$  folgt. □

# Eine Variante von 3-SAT

## Not-All-Equal-SAT ( $\text{NAESAT}$ ):

Gegeben: Eine Formel  $F$  in 3-KNF.

Gefragt: Hat  $F$  eine (erfüllende) Belegung, unter der in keiner Klausel alle Literale denselben Wahrheitswert haben?

## Satz

$\text{NAESAT}$  ist NP-vollständig.

## Beweis

- $\text{NAESAT} \in \text{NP}$  ist klar.
- Wir reduzieren  $\text{CIRSAT}$  auf  $\text{NAESAT}$ , indem wir die Reduktion von  $\text{CIRSAT}$  auf 3-SAT geeignet anpassen.

# NAESAT ist NP-vollständig

## Reduktion von CIRSAT auf NAESAT

- Wir reduzieren CIRSAT auf NAESAT, indem wir die Reduktion von CIRSAT auf 3-SAT geeignet anpassen:

Gatter $g_i$	zugeh. Klauseln in $F'_S$	Semantik
0	$\{\bar{y}_i, z\}$	$y_i = 0$
1	$\{y_i, z\}$	$y_i = 1$
$x_j$	$\{\bar{y}_i, x_j, z\}, \{\bar{x}_j, y_i, z\}$	$y_i \leftrightarrow x_j$
$(\neg, j)$	$\{\bar{y}_i, \bar{y}_j, z\}, \{y_j, y_i, z\}$	$y_i \leftrightarrow \bar{y}_j$
$(\wedge, j, k)$	$\{\bar{y}_i, y_j, z\}, \{\bar{y}_i, y_k, z\}, \{\bar{y}_j, \bar{y}_k, y_i\}$	$y_i \leftrightarrow y_j \wedge y_k$
$(\vee, j, k)$	$\{\bar{y}_j, y_i, z\}, \{\bar{y}_k, y_i, z\}, \{\bar{y}_i, y_j, y_k\}$	$y_i \leftrightarrow y_j \vee y_k$

- Es ist leicht zu sehen, dass die Dreierklauseln unter jeder erfüllenden Belegung von  $F_S$  bereits beide Wahrheitswerte annehmen.
- Zu den übrigen Klauseln können wir eine neue Variable  $z$  hinzufügen.

# NAESAT ist NP-vollständig

## Reduktion von CIRSAT auf NAESAT

- Sei  $F'_s(x_1, \dots, x_n, y_1, \dots, y_m, z)$  die 3-KNF Formel, die die Klausel  $\{y_m, z\}$  und für jedes Gatter  $g_i$  die folgenden Klauseln enthält:

Gatter $g_i$	zugeh. Klauseln $F'_s$
0	$\{\bar{y}_i, z\}$
1	$\{y_i, z\}$
$x_j$	$\{\bar{y}_i, x_j, z\}, \{\bar{x}_j, y_i, z\}$
$(\neg, j)$	$\{\bar{y}_i, \bar{y}_j, z\}, \{y_j, y_i, z\}$
$(\wedge, j, k)$	$\{\bar{y}_i, y_j, z\}, \{\bar{y}_i, y_k, z\}, \{\bar{y}_j, \bar{y}_k, y_i\}$
$(\vee, j, k)$	$\{\bar{y}_j, y_i, z\}, \{\bar{y}_k, y_i, z\}, \{\bar{y}_i, y_j, y_k\}$

- $F'_s$  entsteht also aus  $F_s$ , indem wir zu jeder Klausel mit  $\leq 2$  Literalen die neue Variable  $z$  hinzufügen.

# NAESAT ist NP-vollständig

## Reduktion von CIRSAT auf NAESAT

- Da die Funktion  $f : s \mapsto F'_s$  offenbar in FP berechenbar ist, bleibt nur noch die Korrektheit von  $f$  zu zeigen:

$$s \in \text{CIRSAT} \Leftrightarrow F'_s \in \text{NAESAT}.$$

- Ist  $s \in \text{CIRSAT}$ , so existiert eine Belegung  $ab$  mit  $F_s(ab) = 1$ .
- Folglich enthalten unter dieser Belegung alle Klauseln von  $F_s$  (und damit auch alle Klauseln von  $F'_s$ ) ein wahres Literal.
- Tatsächlich wird unter  $ab$  in jeder Dreierklausel von  $F_s$  (und damit in jeder Klausel von  $F'_s$ , die nicht  $z$  enthält) auch ein Literal falsch, da  $ab$  neben jeder Dreierklausel der Form
  - $\{\bar{y}_i, y_j, y_k\}$  die Klauseln  $\{\bar{y}_j, y_i\}$  und  $\{\bar{y}_k, y_i\}$  und neben
  - $\{y_i, \bar{y}_j, \bar{y}_k\}$  die Klauseln  $\{y_j, \bar{y}_i\}$  und  $\{y_k, \bar{y}_j\}$  erfüllt.
- Setzen wir also  $z = 0$ , so wird in jeder Klausel von  $F'_s$  mindestens ein Literal wahr und mindestens ein Literal falsch.

# NAESAT ist NP-vollständig

## Reduktion von CIRSAT auf NAESAT

- die umgekehrte Implikation sei nun  $F'_s \in \text{NAESAT}$  angenommen.
- Dann existiert eine Belegung  $abc \in \{0, 1\}^{n+m+1}$  für

$$F'_s(x_1, \dots, x_n, y_1, \dots, y_m, z),$$

unter der in jeder Klausel ein wahres und ein falsches Literal vorkommen.

- Da dies auch unter der komplementären Belegung  $\overline{abc}$  der Fall ist, können wir  $c = 0$  annehmen.
- Dann erfüllt aber die Belegung  $ab$  die Formel  $F_s$ .
- Also ist  $s(a) = 1$  und damit  $s \in \text{CIRSAT}$ .



# SAT als Optimierungsproblem

- Ist eine KNF-Formel  $F$  nicht erfüllbar, so kann es nützlich sein, herauszufinden, wieviele Klauseln in  $F$  maximal erfüllbar sind.
- Die Schwierigkeit dieses Optimierungsproblems lässt sich durch folgendes Entscheidungsproblem charakterisieren.

## MAX- $k$ -SAT:

**Gegeben:** Eine Formel  $F$  in  $k$ -KNF und eine Zahl  $l$ .

**Gefragt:** Existiert eine Belegung  $a$ , die mindestens  $l$  Klauseln in  $F$  erfüllt?

Hierbei können Klauseln in  $F$  auch mehrfach vorkommen.

## Satz

- 1 MAX-1-SAT  $\in$  P.
- 2 MAX-2-SAT  $\in$  NPC.

# SAT als Optimierungsproblem

## Reduktion von 3-SAT auf MAX-2-SAT

- Für eine Dreierklausel  $C = \{l_1, l_2, l_3\}$ , in der die Variable  $v$  nicht vorkommt, sei  $G(l_1, l_2, l_3, v)$  die 2-KNF Formel, die aus folgenden 10 Klauseln besteht:

$$\{l_1\}, \{l_2\}, \{l_3\}, \{v\}, \{\bar{l}_1, \bar{l}_2\}, \{\bar{l}_2, \bar{l}_3\}, \{\bar{l}_1, \bar{l}_3\}, \{l_1, \bar{v}\}, \{l_2, \bar{v}\}, \{l_3, \bar{v}\}$$

- Die folgenden 3 Eigenschaften von  $G$  sind leicht zu verifizieren:
  - Keine Belegung von  $G$  erfüllt mehr als 7 Klauseln von  $G$ .
  - Jede Belegung  $a$  von  $C$  mit  $C(a) = 1$  ist zu einer Belegung  $a'$  von  $G$  erweiterbar, die 7 Klauseln von  $G$  erfüllt.
  - Keine Belegung  $a$  von  $C$  mit  $C(a) = 0$  ist zu einer Belegung  $a'$  von  $G$  erweiterbar, die 7 Klauseln von  $G$  erfüllt.
- Sei  $F$  eine 3-KNF-Formel über  $x_1, \dots, x_n$  mit  $m$  Klauseln.
- Wir nehmen an, dass  $C_j = \{l_{j1}, l_{j2}, l_{j3}\}$ ,  $j = 1, \dots, k$ , die Dreier- und  $C_{k+1}, \dots, C_m$  die Einer- und Zweierklauseln von  $F$  sind.



# SAT als Optimierungsproblem

## Reduktion von 3-SAT auf MAX-2-SAT

- Wir nehmen an, dass  $C_j = \{l_{j1}, l_{j2}, l_{j3}\}$ ,  $j = 1, \dots, k$ , die Dreier- und  $C_{k+1}, \dots, C_m$  die Einer- und Zweierklauseln von  $F$  sind.
- Betrachte die 2-KNF Formel  $F'$  mit  $10k + (m - k)$  Klauseln ist, die wie folgt aus  $F$  entsteht:
  - Ersetze jede Dreierklausel  $C_j = \{l_{j1}, l_{j2}, l_{j3}\}$  in  $F$  durch die 10 Klauseln der 2-KNF-Formel  $G_j = G(l_{j1}, l_{j2}, l_{j3}, v_j)$ .
- Dann gilt
$$F \in 3\text{-SAT} \Leftrightarrow (F', m + 6k) \in \text{MAX-2-SAT}.$$
- Die Vorwärtsrichtung ergibt sich unmittelbar aus der Tatsache, dass jede erfüllende Belegung  $a$  für  $F$  zu einer Belegung  $a'$  für  $F'$  erweiterbar ist, die  $7k + m - k = m + 6k$  Klauseln von  $F'$  erfüllt.

# SAT als Optimierungsproblem

## Reduktion von 3-SAT auf MAX-2-SAT

- Dann gilt

$$F \in 3\text{-SAT} \Leftrightarrow (F', m + 6k) \in \text{MAX-2-SAT}.$$

- Die Vorwärtsrichtung ergibt sich unmittelbar aus der Tatsache, dass jede erfüllende Belegung  $a$  für  $F$  zu einer Belegung  $a'$  für  $F'$  erweiterbar ist, die  $7k + m - k = m + 6k$  Klauseln von  $F'$  erfüllt.
- Für die Rückwärtsrichtung sei  $a$  eine Belegung, die mindestens  $m + 6k$  Klauseln von  $F'$  erfüllt.
- Da in jeder 10er-Gruppe  $G_j$ ,  $j = 1, \dots, k$ , maximal 7 Klauseln erfüllbar sind, muss  $a$  in jeder 10er-Gruppe genau 7 Klauseln und zudem alle Klauseln  $C_j$  für  $j = k + 1, \dots, m$  erfüllen.
- Dies ist aber nur möglich, wenn  $a$  alle Klauseln  $C_j$  von  $F$  erfüllt.
- Zudem ist leicht zu sehen, dass  $g : F \mapsto (F', m + 6k)$  in FP berechenbar ist, woraus  $3\text{-SAT} \leq^P \text{MAX-2-SAT}$  folgt.



# Das Wortproblem für NFAs

- Als nächstes zeigen wir, dass das Wortproblem für NFAs effizient entscheidbar ist.
- Dagegen wird sich das Äquivalenzproblem für NFAs als schwer lösbar (genauer: co-NP-hart) herausstellen.

## Satz

Das Wortproblem für NFAs,

$$\text{WP}_{\text{NFA}} = \{N\#x \mid N \text{ ist ein NFA und } x \in L(N)\},$$

ist in P entscheidbar.

## P-Algorithmus für $\text{WP}_{\text{NFA}}$

- 1 **Input:** Ein NFA  $N = (Z, \Sigma, \delta, Q_0, E)$  und ein Wort  $x = x_1 \dots x_n$
- 2  $Q := Q_0$
- 3 **for**  $i := 1$  **to**  $n$  **do**
- 4  $Q := \bigcup_{q \in Q} \delta(q, x_i)$
- 5 **if**  $Q \cap E \neq \emptyset$  **then accept else reject**

# Das Wortproblem für NFAs ist in P entscheidbar

## Satz

Das Wortproblem für NFAs,

$$WP_{\text{NFA}} = \{N \# x \mid N \text{ ist ein NFA und } x \in L(N)\},$$

ist in P entscheidbar.

## P-Algorithmus für $WP_{\text{NFA}}$

```
1 Input: Ein NFA  $N = (Z, \Sigma, \delta, Q_0, E)$  und ein Wort  $x = x_1 \dots x_n$ 
2    $Q := Q_0$ 
3   for  $i := 1$  to  $n$  do
4      $Q := \bigcup_{q \in Q} \delta(q, x_i)$ 
5   if  $Q \cap E \neq \emptyset$  then accept else reject
```

Es ist klar, dass dieser Algorithmus korrekt arbeitet und auch in eine polynomiell zeitbeschränkte DTM für die Sprache  $WP_{\text{NFA}}$  transformiert werden kann.

# Das Leerheitsproblem und das Schnittproblem für NFAs

Ganz ähnlich folgt auch, dass das Leerheits- und das Schnittproblem für NFAs in Polynomialzeit lösbar sind (siehe Übungen).

## Korollar

Das Leerheitsproblem für NFAs,

$$LP_{\text{NFA}} = \{N \mid N \text{ ist ein NFA und } L(N) = \emptyset\}$$

und das Schnittproblem für NFAs,

$$SP_{\text{NFA}} = \{N_1 \# N_2 \mid N_1 \text{ und } N_2 \text{ sind NFAs mit } L(N_1) \cap L(N_2) \neq \emptyset\}$$

sind in P lösbar.

# Entscheidungsprobleme für reguläre Ausdrücke

## Korollar

Das Wort-, das Leerheits- und das Schnittproblem für reguläre Ausdrücke sind in  $P$  entscheidbar.

## Beweis

- Ein regulärer Ausdruck  $\alpha$  lässt sich in Polynomialzeit in einen äquivalenten NFA  $N_\alpha$  transformieren (siehe Übungen).
- Daher gilt  $WP_{RA} \leq^P WP_{NFA}$  mittels  $f : (\alpha \# x) \mapsto (N_\alpha \# x)$ .
- Da nach vorigem Satz  $WP_{NFA} \in P$  ist, und da  $P$  unter  $\leq^P$  abgeschlossen ist, folgt  $WP_{RA} \in P$ .
- Ähnlich zeigt man  $LP_{RA} \leq^P LP_{NFA}$  und  $SP_{RA} \leq^P SP_{NFA}$ , was die Zugehörigkeit von  $LP_{RA}$  und  $SP_{RA}$  zu  $P$  impliziert. □

Dagegen sind das Äquivalenz- und das Inklusionsproblem für reguläre Ausdrücke co-NP-hart. Dies gilt sogar für **sternfreie** reguläre Ausdrücke (kurz SFRAs), also für reguläre Ausdrücke, die keinen Stern enthalten.

# Entscheidungsprobleme für reguläre Sprachen

Die Tabelle gibt die Komplexitäten der wichtigsten Entscheidungsprobleme für durch DFAs, NFAs oder (sternfreie) reguläre Ausdrücke gegebene reguläre Sprachen an.

	Wort- problem $x \in L?$	Leerheits- problem $L = \emptyset?$	Schnitt- problem $L_1 \cap L_2 \neq \emptyset?$	Äquivalenz- problem $L_1 = L_2?$	Inklusions- problem $L_1 \subseteq L_2$
DFA	P	P	P	P	P
SFRA	P	P	P	co-NP-vollständig	
RA	P	P	P	PSPACE-vollständig	
NFA	P	P	P	PSPACE-vollständig	

Als nächstes zeigen wir, dass die Probleme  $\text{ÄP}_{\text{SFRA}}$  und  $\text{IP}_{\text{SFRA}}$  co-NP-vollständig sind.

## Das Äquivalenzproblem für sternfreie reguläre Ausdrücke

### Lemma

Folgende Sprache ist NP-vollständig:

$$SF = \{\alpha\#0^n \mid \alpha \text{ ist ein SFRA über } \{0,1\} \text{ mit } L(\alpha) \neq \{0,1\}^n\}.$$

### Beweis

- Wir zeigen zuerst  $SF \in NP$ .
- Anschließend reduzieren wir 3-SAT auf SF.



## Das Äquivalenzproblem für sternfreie reguläre Ausdrücke

- Sei  $\alpha$  ein sternfreier regulärer Ausdruck der Länge  $m$ .
- Es ist leicht zu zeigen (durch Induktion über den Aufbau von  $\alpha$ ), dass  $L(\alpha)$  nur Wörter der Länge  $\leq m$  enthält.
- D.h. es gilt  $L(\alpha) \subseteq \{0, 1\}^{\leq m}$ , wobei  $\Sigma^{\leq m} = \bigcup_{i=0}^m \Sigma^i$  ist.
- Daher akzeptiert folgender NP-Algorithmus die Sprache  $SF = \{\alpha \# 0^n \mid \alpha \text{ ist SFRA über } \{0, 1\} \text{ mit } L(\alpha) \neq \{0, 1\}^n\}$ .

### NP-Algorithmus für SF

```
1  Input: SFRA  $\alpha$  über  $\{0, 1\}$  und eine unär kodierte Zahl  $n$ 
2  guess  $x \in \{0, 1\}^n$ 
3  if  $x \notin L(\alpha)$  then accept
4   $m := |\alpha|$ 
5  guess  $y \in \{0, 1\}^{\leq m}$ 
6  if  $y \in L(\alpha) - \{0, 1\}^n$  then accept else reject
```

# Das Äquivalenzproblem für sternfreie reguläre Ausdrücke

## Reduktion von 3-SAT auf SF

- Sei  $F = \{C_1, \dots, C_m\}$  eine 3-KNF Formel.
- Betrachte den SFRA  $\alpha_F = (\alpha_1 | \dots | \alpha_m)$  mit  $\alpha_j = \beta_{j1} \cdots \beta_{jn}$  und

$$\beta_{ji} = \begin{cases} 0, & x_i \in C_j, \\ 1, & \bar{x}_i \in C_j, \\ (0|1), & \text{sonst.} \end{cases}$$

- Dann ist  $L(\alpha_j) = \{a \in \{0, 1\}^n \mid C_j(a) = 0\}$  und daher folgt

$$\begin{aligned} F \in 3\text{-SAT} &\Leftrightarrow \exists a \in \{0, 1\}^n : F(a) = 1 \\ &\Leftrightarrow \exists a \in \{0, 1\}^n \forall j = 1, \dots, m : C_j(a) = 1 \\ &\Leftrightarrow \exists a \in \{0, 1\}^n \forall j = 1, \dots, m : a \notin L(\alpha_j) \\ &\Leftrightarrow \exists a \in \{0, 1\}^n : a \notin L(\alpha_F) \\ &\Leftrightarrow L(\alpha_F) \neq \{0, 1\}^n. \end{aligned}$$

## Das Äquivalenzproblem für sternfreie reguläre Ausdrücke

### Korollar

Das Äquivalenzproblem für sternfreie reguläre Ausdrücke,

$$\text{ÄP}_{\text{SFRA}} = \{\alpha\#\beta \mid \alpha, \beta \text{ sind SFRA's mit } L(\alpha) = L(\beta)\},$$

ist co-NP-vollständig.

### Beweis

- Wir zeigen, dass das Inäquivalenzproblem für SFRA's NP-vollständig ist.
- Die Zugehörigkeit zu NP liefert folgender Algorithmus:

```
1  Input: SFRA's  $\alpha$  und  $\beta$ 
2     $m := \max\{|\alpha|, |\beta|\}$ 
3    guess  $x \in \{0, 1\}^{\leq m}$ 
4    if  $x \in L(\alpha) \Delta L(\beta)$  then accept else reject
```

- Zudem lässt sich SF leicht auf dieses Problem reduzieren:

$$\alpha\#0^n \mapsto \alpha\# \underbrace{(0|1) \cdots (0|1)}_{n\text{-mal}}.$$

## Das Äquivalenz- und das Inklusionsproblem für NFAs

- Das Komplement von SF lässt sich auch auf das Inklusionsproblem für sternfreie reguläre Ausdrücke reduzieren (siehe Übungen).
- Folglich ist  $IP_{SFRA}$  ebenfalls co-NP-vollständig.
- Das Äquivalenz- und Inklusionsproblem für reguläre Ausdrücke sind demnach co-NP-hart.
- Diese Probleme sind sogar PSPACE-vollständig (auch für NFAs).

	Wort- problem $x \in L?$	Leerheits- problem $L = \emptyset?$	Schnitt- problem $L_1 \cap L_2 \neq \emptyset?$	Äquivalenz- problem $L_1 = L_2?$	Inklusions- problem $L_1 \subseteq L_2$
DFA	P	P	P	P	P
SFRA	P	P	P	co-NP-vollständig	
RA	P	P	P	PSPACE-vollständig	
NFA	P	P	P	PSPACE-vollständig	

## Notation – ungerichtete Graphen

- Ein (**ungerichteter**) **Graph** ist ein Paar  $G = (V, E)$ , wobei
  - $V$  - eine endliche Menge von **Knoten/Ecken** und
  - $E$  - die Menge der **Kanten** ist.

Hierbei gilt

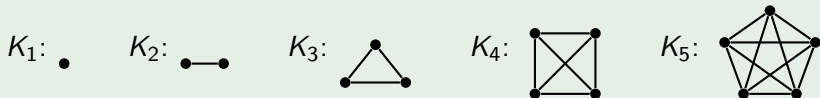
$$E \subseteq \binom{V}{2} := \{\{u, v\} \subseteq V \mid u \neq v\}.$$

- Die **Knotenzahl** von  $G$  ist  $n(G) = \|V\|$ .
- Die **Kantenzahl** von  $G$  ist  $m(G) = \|E\|$ .
- Die **Nachbarschaft** von  $v \in V$  ist  $N_G(v) = \{u \in V \mid \{u, v\} \in E\}$  und die Nachbarschaft von  $U \subseteq V$  ist  $N_G(U) = \bigcup_{u \in U} N_G(u)$ .
- Der **Grad** von  $v \in V$  ist  $\deg_G(v) = \|N_G(v)\|$ .
- Der **Minimalgrad** von  $G$  ist  $\delta(G) := \min_{v \in V} \deg_G(v)$  und
- der **Maximalgrad** von  $G$  ist  $\Delta(G) := \max_{v \in V} \deg_G(v)$ .

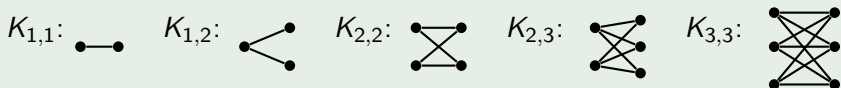
Falls  $G$  aus dem Kontext ersichtlich ist, schreiben wir auch einfach  $n$ ,  $m$ ,  $N(v)$ ,  $\deg(v)$ ,  $\delta$  usw.

## Beispiel

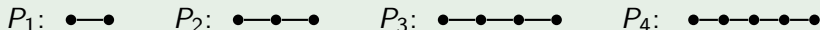
- Der **vollständige Graph**  $(V, E)$  mit  $\|V\| = n$  und  $E = \binom{V}{2}$  wird mit  $K_n$  und der **leere Graph**  $(V, \emptyset)$  wird mit  $E_n$  bezeichnet.



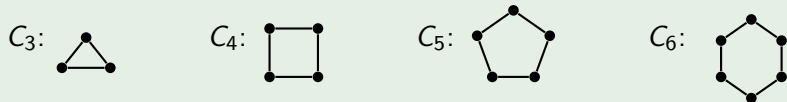
- Der **vollständige bipartite Graph**  $(A, B, E)$  auf  $a + b$  Knoten, d.h.  $A \cap B = \emptyset$ ,  $\|A\| = a$ ,  $\|B\| = b$  und  $E = \{\{u, v\} \mid u \in A, v \in B\}$  wird mit  $K_{a,b}$  bezeichnet.



- Der **Pfad der Länge  $n$**  wird mit  $P_n$  bezeichnet.



- Der **Kreis der Länge  $n$**  wird mit  $C_n$  bezeichnet.



## Notation – ungerichtete Graphen (Fortsetzung)

- Ein Graph  $H = (V', E')$  heißt **Sub-/Teil-/Untergraph** von  $G = (V, E)$ , falls  $V' \subseteq V$  und  $E' \subseteq E$  ist.

Der Subgraph  $H$  heißt (**durch  $V'$** ) **induziert**, falls  $E' = E \cap \binom{V'}{2}$  ist. Hierfür schreiben wir auch  $H = G[V']$ .

- Ein **Weg** ist eine Folge von Knoten  $v_0, \dots, v_j$  mit  $\{v_i, v_{i+1}\} \in E$  für  $i = 0, \dots, j - 1$ .

Ein Weg heißt **einfach** oder **Pfad**, falls alle durchlaufenen Knoten verschieden sind.

Die **Länge** des Weges ist die Anzahl der Kanten, also  $j$ .

Ein Weg  $v_0, \dots, v_j$  heißt auch  **$v_0$ - $v_j$ -Weg**.

- Ein **Zyklus** ist ein  $u$ - $v$ -Weg der Länge  $j \geq 2$  mit  $u = v$ .
- Ein **Kreis** ist ein Zyklus  $v_0, v_1, \dots, v_{j-1}, v_0$  der Länge  $j \geq 3$ , für den  $v_0, v_1, \dots, v_{j-1}$  paarweise verschieden sind.

# Cliquen, Stabilität und Kantenüberdeckungen

- Eine Knotenmenge  $U \subseteq V$  heißt **Clique**, wenn jede Kante mit beiden Endpunkten in  $U$  in  $E$  ist, d.h. es gilt  $\binom{U}{2} \subseteq E$ .

Die **Cliquenzahl** ist

$$\omega(G) := \max\{\|U\| \mid U \text{ ist Clique in } G\}.$$

- Eine Knotenmenge  $U \subseteq V$  heißt **stabil**, wenn keine Kante in  $G$  beide Endpunkte in  $U$  hat, d.h. es gilt  $E \cap \binom{U}{2} = \emptyset$ .

Die **Stabilitätszahl** ist

$$\alpha(G) := \max\{\|U\| \mid U \text{ ist stabile Menge in } G\}.$$

- Eine Knotenmenge  $U \subseteq V$  heißt **Kantenüberdeckung** (engl. *vertex cover*), wenn jede Kante  $e \in E$  mindestens einen Endpunkt in  $U$  hat, d.h. es gilt  $e \cap U \neq \emptyset$  für alle Kanten  $e \in E$ .

Die **Überdeckungsanzahl** ist

$$\beta(G) = \min\{\|U\| \mid U \text{ ist eine Kantenüberdeckung in } G\}.$$



# Fundamentale Graphprobleme

Für einen gegebenen Graphen  $G$  und eine gegebene Zahl  $k \geq 1$  betrachten wir die folgenden Fragestellungen:

**CLIQUE:**

Gefragt: Hat  $G$  eine Clique der Größe  $k$ ?

**INDEPENDENT SET (IS):**

Gefragt: Hat  $G$  eine stabile Menge der Größe  $k$ ?

**VERTEX COVER (VC):**

Gefragt: Hat  $G$  eine Kantenüberdeckung der Größe  $k$ ?

**Satz**

CLIQUE, IS und VC sind NP-vollständig.

# IS ist NP-vollständig

## Reduktion von 3-SAT auf IS

- Sei  $F = \{C_1, \dots, C_m\}$  mit  $C_j = \{l_{j,1}, \dots, l_{j,k_j}\}$  für  $j = 1, \dots, m$  eine 3-KNF-Formel über den Variablen  $x_1, \dots, x_n$ .

- Betrachte den Graphen  $G = (V, E)$  mit

$$V = \{v_{ji} \mid 1 \leq j \leq m, 1 \leq i \leq k_j\} \text{ und}$$

$$E = \{\{v_{ji}, v_{j'i'}\} \in \binom{V}{2} \mid j = j' \text{ oder } l_{ji} = \bar{l}_{j'i'} \text{ oder } l_{j'i'} = \bar{l}_{ji}\}.$$

- Nun gilt

$F \in 3\text{-SAT} \Leftrightarrow$  es gibt eine Belegung, die in jeder Klausel  $C_j$  (mindestens) ein Literal  $l_{j,i_j}$  wahr macht

$\Leftrightarrow$  es gibt  $m$  Literale  $l_{1,i_1}, \dots, l_{m,i_m}$ , die paarweise nicht komplementär sind (d.h.  $l_{j,i_j} \neq \bar{l}_{j',i_{j'}}$  für  $j \neq j'$ )

$\Leftrightarrow$  es gibt  $m$  Knoten  $v_{1,i_1}, \dots, v_{m,i_m}$ , die nicht durch Kanten verbunden sind

$\Leftrightarrow G$  besitzt eine stabile Menge von  $m$  Knoten. □

# CLIQUE ist NP-vollständig

## Korollar

CLIQUE ist NP-vollständig.

## Beweis

- Es ist leicht zu sehen, dass jede Clique in einem Graphen  $G = (V, E)$  eine stabile Menge in dem zu  $G$  komplementären Graphen  $\bar{G} = (V, \bar{E})$  mit  $\bar{E} = \binom{V}{2} \setminus E$  ist und umgekehrt.
- Daher lässt sich IS mittels

$$f : (G, k) \mapsto (\bar{G}, k)$$

auf CLIQUE reduzieren. □

# VC ist NP-vollständig

## Korollar

VC ist NP-vollständig.

## Beweis

- Offensichtlich ist eine Menge  $I$  genau dann stabil, wenn ihr Komplement  $V \setminus I$  eine Kantenüberdeckung ist.
- Daher lässt sich IS mittels

$$f : (G, k) \mapsto (G, n(G) - k)$$

auf VC reduzieren. □

# Färbbarkeit von Graphen

## Definition

- Eine Abbildung  $f: V \rightarrow \mathbb{N}$  heißt **Färbung** von  $G$ , wenn  $f(u) \neq f(v)$  für alle  $\{u, v\} \in E$  gilt.
- $G$  heißt  **$k$ -färbbar**, falls eine Färbung  $f: V \rightarrow \{1, \dots, k\}$  existiert.
- Die **chromatische Zahl** ist

$$\chi(G) = \min\{k \in \mathbb{N} \mid G \text{ ist } k\text{-färbbar}\}.$$

## $k$ -Färbbarkeit ( $k$ -COLORING):

Gegeben: Ein Graph  $G$ .

Gefragt: Ist  $G$   $k$ -färbbar?

## Satz

- 1-COLORING, 2-COLORING  $\in P$  (siehe Übungen).
- 3-COLORING ist NP-vollständig.

# Färbbarkeit von Graphen

## Reduktion von NAESAT auf 3-COLORING

- Sei eine 3-KNF-Formel  $F = \{C_1, \dots, C_m\}$  über den Variablen  $x_1, \dots, x_n$  mit Klauseln

$$C_j = \{l_{j,1}, \dots, l_{j,k_j}\}, \quad k_j \leq 3$$

gegeben.

- Wir können annehmen, dass  $F$  keine Einerklauseln enthält.
- Wir konstruieren einen Graphen  $G_F = (V, E)$ , der genau dann 3-färbbar ist, wenn  $F \in \text{NAESAT}$  ist.
- Wir setzen

$$V = \{s, x_1, \dots, x_n, \bar{x}_1, \dots, \bar{x}_n\} \cup \{v_{jk} \mid 1 \leq j \leq m, 1 \leq k \leq k_j\}$$

und

$$E = \left\{ \{s, x_i\}, \{s, \bar{x}_i\}, \{x_i, \bar{x}_i\} \mid 1 \leq i \leq n \right\} \cup \left\{ \{s, v_{jk}\} \mid k_j = 2 \right\} \cup \\ \left\{ \{v_{jk}, v_{jl}\} \mid k \neq l \right\} \cup \left\{ \{v_{jk}, x_i\} \mid l_{jk} = \bar{x}_i \right\} \cup \left\{ \{v_{jk}, \bar{x}_i\} \mid l_{jk} = x_i \right\}.$$

# Färbbarkeit von Graphen

## Reduktion von NAESAT auf 3-COLORING

- Wir setzen

$$V = \{s\} \cup \{x_1, \dots, x_n, \bar{x}_1, \dots, \bar{x}_n\} \cup \{v_{jk} \mid 1 \leq j \leq m, 1 \leq k \leq k_j\},$$

und

$$E = \left\{ \{s, x_i\}, \{s, \bar{x}_i\}, \{x_i, \bar{x}_i\} \mid 1 \leq i \leq n \right\} \cup \left\{ \{s, v_{jk}\} \mid k_j = 2 \right\} \cup \\ \left\{ \{v_{jk}, v_{jl}\} \mid k \neq l \right\} \cup \left\{ \{v_{jk}, x_i\} \mid l_{jk} = \bar{x}_i \right\} \cup \left\{ \{v_{jk}, \bar{x}_i\} \mid l_{jk} = x_i \right\}.$$

- Sei  $a = a_1 \cdots a_n$  eine Belegung für  $F$ , unter der in jeder Klausel  $C_j = \{l_{j1}, \dots, l_{jk_j}\}$  ein Literal wahr und eines falsch wird.
- Wir können annehmen, dass  $l_{j1}(a) = 0$  und  $l_{j2}(a) = 1$  ist.
- Dann lässt sich  $G_F$  wie folgt mit den 3 Farben 0, 1, 2 färben:

Knoten $v$	$s$	$x_i$	$\bar{x}_i$	$v_{j1}$	$v_{j2}$	$v_{j3}$ (falls $k_j = 3$ )
Farbe $c(v)$	2	$a_i$	$\bar{a}_i$	0	1	2

# Färbbarkeit von Graphen

## Reduktion von NAESAT auf 3-COLORING

- Sei nun umgekehrt  $c : V \rightarrow \{0, 1, 2\}$  eine 3-Färbung von  $G_F$ .
- Dann können wir annehmen, dass  $c(v) = 2$  ist.
- Also gilt für  $i = 1, \dots, n$ , dass  $\{c(x_i), c(\bar{x}_i)\} = \{0, 1\}$  ist.
- Zudem müssen die Knoten  $v_{j_1}, \dots, v_{j_{k_j}}$  im Fall  $k_j = 2$  mit 0 und 1 und im Fall  $k_j = 3$  mit allen drei Farben 0, 1 und 2 gefärbt sein.
- Wir können annehmen, dass  $c(v_{j_1}) = 0$  und  $c(v_{j_2}) = 1$  ist.
- Wegen  $\{v_{jk}, \bar{l}_{jk}\} \in E$  muss  $c(v_{jk}) \neq c(\bar{l}_{jk})$  für  $k = 1, \dots, k_j$  und daher  $c(v_{jk}) = c(l_{jk})$  für  $k = 1, 2$  gelten.
- Also macht die Belegung  $a = c(x_1) \cdots c(x_n)$  die Literale  $l_{j_1}$  falsch und  $l_{j_2}$  wahr.
- Insgesamt gilt also

$$F \in \text{NAESAT} \Leftrightarrow G_F \in \text{3-COLORING.}$$





# Matchings und der Heiratsatz

## Das Heiratsproblem

- Sei  $V$  eine Gruppe von heiratswilligen Personen.
- Wir verbinden  $u, w \in V$  durch eine Kante, falls aus Sicht von  $u$  und  $w$  die Möglichkeit einer Heirat zwischen  $u$  und  $w$  besteht.
- Sind Viehlen ausgeschlossen, so lässt sich jedes mögliche Heiratsarrangement durch ein Matching beschreiben.

## Definition

Sei  $G = (V, E)$  ein Graph.

- Zwei Kanten  $e, e' \in E$  heißen **unabhängig**, falls  $e \cap e' = \emptyset$  ist.
- Eine Kantenmenge  $M \subseteq E$  heißt **Matching** in  $G$ , falls die Kanten in  $M$  paarweise unabhängig sind.

# Matchings und der Heiratssatz

## Definition

Sei  $G = (V, E)$  ein Graph.

- Zwei Kanten  $e, e' \in E$  heißen **unabhängig**, falls  $e \cap e' = \emptyset$  ist.
- Eine Kantenmenge  $M \subseteq E$  heißt **Matching** in  $G$ , falls die Kanten in  $M$  paarweise unabhängig sind.
- Die **Matchingzahl** von  $G$  ist

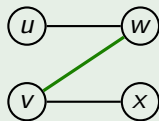
$$\mu(G) = \max\{\|M\| \mid M \text{ ist ein Matching in } G\}.$$

- Ein Matching  $M$  der Größe  $\|M\| = \mu(G)$  heißt **optimal**.
- $M$  heißt **gesättigt**, falls  $M \cup \{e\}$  für keine Kante  $e \in E \setminus M$  ein Matching ist.
- Analog lassen sich Matchings  $M \subseteq E$  in einem  **$d$ -uniformen Hypergraphen**  $G = (V, E)$  mit  $E \subseteq \binom{V}{d}$  definieren.
- Ein Matching  $M$  heißt **perfekt**, falls  $\|M\| \geq \lfloor \|V\|/d \rfloor$  ist.

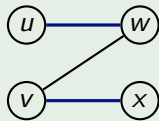
# Matchings und der Heiratssatz

## Beispiel

Ein gesättigtes Matching muss nicht optimal sein:



$M$



$M'$

- Das Matching  $M = \{\{v, w\}\}$  ist gesättigt, da es sich nicht zu einem größeren Matching erweitern lässt.
- $M$  ist jedoch nicht optimal, da  $M' = \{\{u, w\}, \{v, x\}\}$  größer ist.
- Die Greedy-Methode, ausgehend von  $M = \emptyset$  solange Kanten zu  $M$  hinzuzufügen, bis sich  $M$  nicht mehr zu einem größeren Matching erweitern lässt, funktioniert also i.a. nicht.

## Matchings und der Heiratssatz

Falls beim Heiratsproblem Homoeen ausgeschlossen sind, ist der resultierende Graph  $G = (V, E)$  bipartit.

### Definition

- Sei  $G = (V, E)$  ein Graph und seien  $U, W \subseteq V$ . Dann bezeichne

$$E(U, W) = \left\{ \{u, w\} \in E \mid u \in U, w \in W \right\}$$

die Menge aller Kanten zwischen  $U$  und  $W$ .

- $G$  heißt **bipartit**, falls sich  $V$  in zwei Mengen  $U$  und  $W$  mit  $E(U, W) = E$  zerlegen lässt.
- In diesem Fall notieren wir  $G$  auch in der Form  $G = (U, W, E)$ .

# Matchingprobleme

## $d$ -dimensionales Matching ( $d$ -MATCH)

Gegeben: Ein  $d$ -uniformer Hypergraph  $G = (V, E)$  und eine Zahl  $k \geq 1$ .

Gefragt: Gibt es in  $G$  ein Matching  $M$  der Größe  $\|M\| \geq k$ ?

## Bipartites Matching (BIMATCH)

Gegeben: Ein bipartiter Graph  $G = (U, W, E)$  und eine Zahl  $k \geq 1$ .

Gefragt: Gibt es in  $G$  ein Matching  $M$  der Größe  $\|M\| \geq k$ ?

## Satz

- BIMATCH und  $d$ -MATCH sind für  $d \leq 2$  in P entscheidbar.
- Für  $d \geq 3$  ist  $d$ -MATCH NP-vollständig.