# Automaten und Sprachen | AutoSpr

## Inhalt

S. 4 Reguläre Sprachen

S. 9 Regulären Ausdruck für finden

Ist die Sprache regulär? auch «soll mit einem regulären Ausdruck geprüft werden, …»

JA: **S. 4** DEA, **S. 6** NEA, **S. 8** regulärer Ausdruck

NEIN: **S. 6** Pumping Lemma1

S. 10 Kontextfreie Sprachen

S. 11 kontextfreie Grammatik in Chomksy Normalform finden

Ist die Sprache kontextfrei? Können Sie eine kontextfreie Grammatik angeben? *(1 Ja, 1 Nein)*

JA: **S. 12** Stackautomat, **S. 11** Chomksy Normalform, Grammatik

NEIN: **S. 14** Pumping Lemma 2

S. 14 Turing-Maschine

S. 14 Zustandsdiagramm analysieren & Lösungen am Ende

S. 15 Wie lange braucht eine TM bis zum Stop? Berechnungsgeschichte

S. 16 Entscheidbarkeit

Kann Problem von einem Computer gelöst werden? → Halteproblem/Haltetheorem (Reduktion z.B. mit Debugger) Nichtentscheidbarkeit & Turing-Vollständigkeit der Sprache erwähnen

Kann ein Programm entscheiden, ob die zulässigen Inputs eines beliebigen Programms M eine bestimmte Eigenschaft haben?

NEIN: **S. 17** Satz von Rice

S. 18 Komplexität

S. 19 Kann eine nicht deterministische Turing-Maschine …? → NP-Verifizierer

**S. 20** Warum skaliert Problem nicht? → 1-zu-1 Reduktion auf ein NP vollständiges Problem, Karp

S. 26 Turing-Vollständig

**S. 26** Ist eine Sprache Turing-vollständig, d.h. unbegrenzter grosser Speicher und unendliche Schleife (z.B. auch bei Rekursion) möglich

JA: **S. 16** Turing-Maschinen-Simulator in dieser Sprache

NEIN: Haltetheorem

Ablauf Prüfungsaufgaben

1. eine reguläre Sprache **S. 4**

2. eine nichtreguläre Sprache (Pumping Lemma) **S. 6**

3. eine kontextfreie Sprache (oft mit CNF) **S. 10**

4. eine nicht kontextfreie Sprache (Pumping Lemma) **S. 14**

5. ein nicht entscheidbares Problem (Rice oder Halteproblem) **S. 17**

6. ein Problem in NP (polynomieller Verifizierer) **S. 19**

7. ein NP-vollständiges Problem (polynomielle Reduktion) **S. 20**

8. etwas über Turing Maschinen **S. 14**

## Logik & Quantoren

### **Junktoren / Verknüpfungen**

|  |  |
| --- | --- |
| **¬** Negation nicht **∧** Konjunktion und **∨** Disjunktion oder | **⇒** Implikation wenn … dann …**¬** **⇔** Äquivalenz … genau dann, wenn … |

### Quantoren

Bei Aussagen, die Variablen enthalten, sind 2 Fälle besonders wichtig:

* Allquantor: «für alle n aus den natürlichen Zahlen gilt…»
* Existenzquantor: «es gibt eine natürliche Zahl n, für die gilt...»

### Beweisen

Ein Beweis zeigt die Gültigkeit einer Behauptung unter gewissen Voraussetzungen und Annahmen:

* Direkter Beweis:
* Indirekter Beweis:
* Widerspruchsbeweis:
* Vollständige Induktion:
* Vollständige Fallunterscheidung: Wenn und es gilt, dass  
   ist, dann gilt auch
* Schubfachprinzip: Wenn n + 1 Gegenstände (Objekte) auf n Schubladen (Kategorien) verteilt werden, dann befinden sich in mindestens einem Schubfach 2 Gegenstände.
* Diagonalverfahren (Georg Cantor): Anzahl rationaler Zahlen ist genau so gross wie die Anzahl natürlicher Zahlen.

## Mengenlehre

«Primitive Datentypen der Mathematik», Zusammenfassung bestimmter Objekte.  
Konstruktion: Grundmenge , Prädikat ,

Ein Bild, das Diagramm, Schrift, Reihe, Text enthält.

Automatisch generierte Beschreibung

### Paare, Tupel & Graphen

* Tupel: Zusammengesetzte Datentypen der Mathematik
* Paare (2-Tupel): sind Mengen, Menge aller Paare:
* N-Tupel: Mengen , Menge aller n-Tupel

## sprachen

### alphabet und wort

Eine nichtleere endliche Menge ∑ heisst Alphabet. Die Elemente von ∑ heissen Zeichen.

Wort

Eine Zeichenkette der Länge n ist ein n-Tupel in . Ein Element von heisst   
«Wort der Länge n».

Abgekürzte Schreibweisen:

Leeres Wort

Die Zeichenkette der Länge 0 heisst das leere Wort.

Menge aller Wörter

### Wortlänge

Wörter haben immer endliche Länge.

* Länge des Wortes w:
* Anzahl Zeichen im Wort w: Sei und , dann ist die Anzahl Zeichen im Wort w.

Beispiele

### Sprache

Eine Sprache besteht aus diesen Wörtern. Eine Teilmenge von ist eine Sprache.

Beispiele

* , die leere Sprache
* , Sprache aller Binärstrings :
* , => Alle Wörter beinhalten gleich viele Einsen und Nullen

### Sprachen und Maschinen

Der Compiler entscheidet, welche Speicherketten zur Sprache gehören.

M eine «Maschine»: w wird von M akzeptiert

**Beispiel:** , wird vom Java-Compiler akzeptiert

Zu jeder Maschine gibt es eine Sprache.

## Deterministische endliche automaten (DEA) und reguläre sprachen

### Ein Bild, das Text, Uhr enthält. Automatisch generierte BeschreibungEin Bild, das Tisch, Kalender enthält. Automatisch generierte BeschreibungDefinition

* Zustände:
* Alphabet:
* Übergangsfunktion:
* Startzustand:
* Akzeptierzustände:

Achtung: Nach Erstellung immer nachkontrollieren, ob der DEA vollständig ist!

### Akzeptierte Sprache eines DEA

Der DEA akzeptiert das Wort , wenn er A vom Startzustand in einen Akzeptierzustand überführt.

Die von A akzeptierte Sprache ist

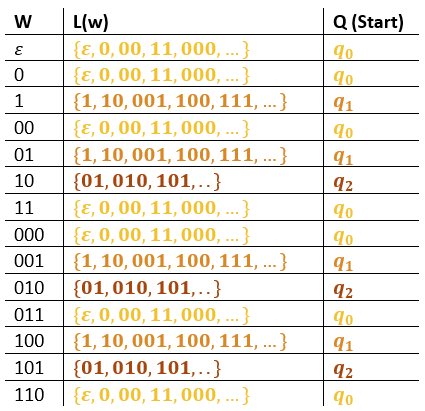
Die Sprache heisst regulär, wenn es einen DEA A gibt, mit .

### Beispiel: Durch drei Teilbare Binärzahlen

* Zustände: Rest bei Teilung durch 3
* Übergänge: ein Bit anhängen (Shift) = Multiplikation mit 2

Ein Bild, das Text, Uhr, Messgerät enthält.

Automatisch generierte Beschreibung

* + 1. Zustandsmenge aus der Sprache ableiten

Wie findet man heraus, wie viele Zustände ein DEA wirklich braucht?

Bei diesem Beispiel:

Alle Werte durchprobieren, in Tabelle schreiben. Es wird Wiederholungen geben. Jede Wiederholung ist der Gleiche Zustand.

Daraus lässt sich der DEA generieren.

Myhill-Nerode Automat

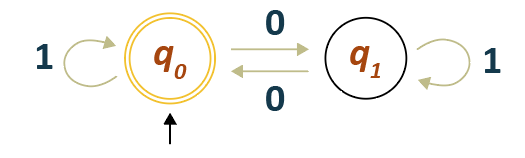
Gestattet es uns, zu einer beliebigen regulären Sprache den endlichen Automaten zu konstruieren. Nicht immer praktisch, da man zum Teil mit grossen Mengen hantieren muss.

Auch nützlich, um redundante Zustände in einem bestehenden DEA zu finden.  
Für ein Wort setze . Insbesondere .

Gegeben: reguläre Sprache L über .Rekonstruiere A mit:

Beispiel:

|  |  |  |
| --- | --- | --- |
| W | L(w) | Q |
|  |  |  |
| 0 |  |  |
| 1 |  |  |
| … | … | … |

Ergibt folgenden Automaten

Eine nicht-reguläre Sprache benötigt unendliche Zustände. Deshalb lässt sich für eine nicht-reguläre Sprache kein endlicher Automat erstellen.

Alternative Art, einen Automaten zu vereinfachen

Zustandspaare finden, die sich nicht zusammenlegen lassen.

Ein Bild, das Diagramm enthält.

Automatisch generierte Beschreibung

«Kreuzchenalgorithmus» - Minimalautomat finden

Dient als Beweis, ob zwei Automaten die gleiche Sprache akzeptieren.

Tabelle erstellen, in der erkennbar ist, ob sich zwei Zustände unterscheiden. Ablauf:

* Alle Zustände zu sich selber äquivalent markieren:
* Paare aus einem Akzeptier- und einem Nichtakzeptierzustand (Zwei Zustände können nicht gleich sein, wenn einer davon ein Akzeptierzustand ist, der andere aber nicht):
* Zweifingertechnik: Von einem Paar alle Übergänge testen, führt zu einem Paar mit ? Falls ja – ebenfalls ein eintragen. Wiederholen für alle Paare ohne Zuordnung.  
  Bsp. und : führt mit einem 1-Übergang zu einem Akzeptierzustand, hingegen nicht.
* Restliche Zustände sind Äquivalent und können minimiert werden.

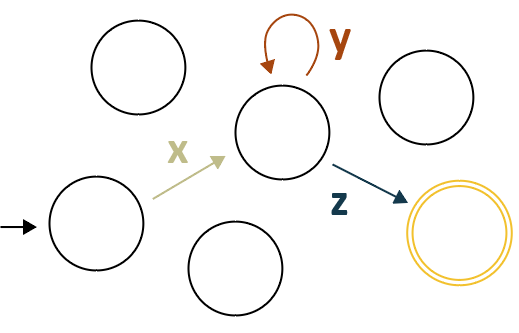
Ein Bild, das Diagramm, Reihe, Schrift, Screenshot enthält.

Automatisch generierte Beschreibung

## Pumping lemma, nea, Mengenoperationen

### Pumping Lemma

Wenn ein Wort länger sein kann als die Anzahl Zustände, dann muss mindestens ein Zustand mehrmals vorkommen im endlichen Automaten: Ein Wort ist aufpumpbar (oder abpumpbar). Man kann damit nachweisen, dass eine Sprache nicht regulär ist, wenn sie mindestens ein Wort enthält, das nicht aufpumpbar ist. Widerspruchsbeweis.

Ist eine reguläre Sprache, dann gibt es , die pumping length so, dass jedes Wort mit in drei Teile zerlegt werden kann mit:

* *also xy auf linker Seite von N, x und z dürfen 0 sein*
  + 1. Widerspruch Beweis/Ablauf (beweisen das Sprache NICHT regulär)

1. Annahme: L ist regulär
2. Es gibt die Pumping Length
3. Wähle ein Wort mit , Definition mit N (nicht mit n) schreiben
4. Aufteilung des Wortes gemäss Pumping Lemma
5. Auswirkung des Pumpens mit Begründung
6. Widerspruch und Schlussfolgerung

Beispiele

* Annahme: L ist regulär
* Pumping Length:
* , = Pumping Length
* Ein Bild, das Text, Screenshot, Reihe, parallel enthält.

  Automatisch generierte BeschreibungAufteilung :

Zuerst N aufteilen und dazwischen die Aktionen / Zeichen/Buchstaben schreiben. Dann xyz schreiben

* Pumpen erhöht nur die Anzahl 0, jedoch nicht Anzahl 1
* Widerspruch: , für , L ist also nicht regulär
* Annahme: L ist regulär
* Pumping Length:
* , = Pumping Length
* Ein Bild, das Reihe, Screenshot, parallel, Diagramm enthält.

  Automatisch generierte BeschreibungAufteilung

Zuerst N aufteilen und dazwischen die Aktionen / Zeichen/Buchstaben schreiben. Dann xyz schreiben

* Pumpen: Auf der linken Seite des Gleichheitszeichens wird eine grössere Binärzahl wie auf der rechten Seite stehen.
* Widerspruch: L ist nicht regulär

### Nichtdeterministische endliche Automaten

Ein DEA kann sich aufhängen. Jede Verzweigung muss einzeln geprüft werden.

* Zustände:
* Alphabet:
* Ein Bild, das Schrift, Kreis, Text, Reihe enthält.

  Automatisch generierte BeschreibungÜbergangsfunktion:
* Startzustand:
* Akzeptierzustände:

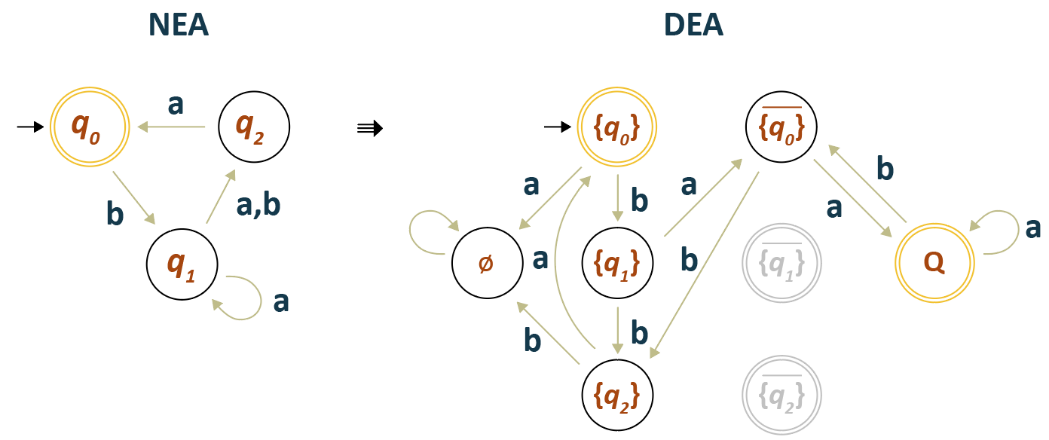
**Beispiel:**

Definition Akzeptieren

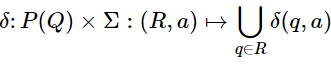
Ein NEA A akzeptiert das Wort , wenn es eine Wahl von Übergängen gibt, derart, dass das Wort w den Automaten in einen Akzeptierzustand überführt.

Faustregel

Nur genau diejenigen Pfeile einzeichnen, die man zum Akzeptieren braucht.

* + 1. Thompson-NEA: Transformation von NEA zu DEA («Könnte»-Automat)
* Alle Kombinationszustände einzeichnen
* alle Zustände mit einem enthaltenen Akzeptierzustand sind Akzeptierzustände
* Vom Startzustand ausgehend alle Übergänge einzeichnen

DEA hat die Potenzmenge der Zustände vom NEA also P(Q).



ℰ-Übergänge Können ohne Verarbeitung eines Zeichens genommen werden. Sind gratis! Jeder NEAℰ kann in einen NEA umgewandelt werden.

Ein Bild, das Kreis, Diagramm, Screenshot, Design enthält.

Automatisch generierte BeschreibungBeispiel Thompson NEAℰ zu DEA

Ein Bild, das Kreis, Diagramm, Screenshot enthält.

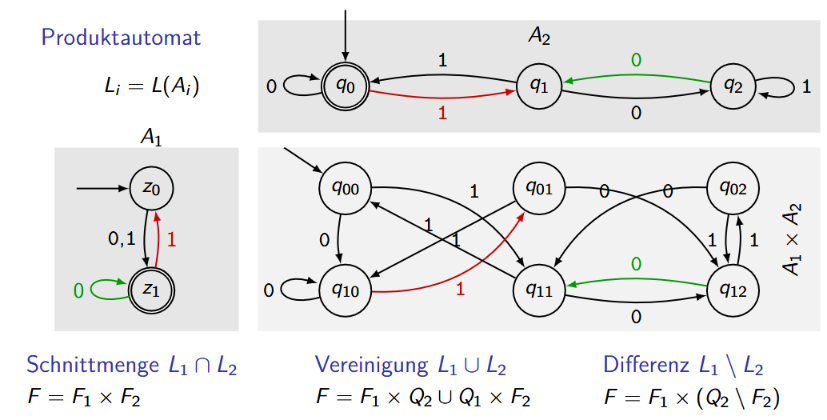
Automatisch generierte Beschreibung

Bild Produktautomat für die nächste Seite:

### Produktautomat

Beispiel im Bild ist die Schnittmenge

* Beide Automaten auf eine Linie strecken / dehnen (falls möglich)
* Alle Übergänge einzeln eintragen, Zustandswechsel des einten Automaten wirkt sich auf die Horizontale aus, die des andern auf die Vertikale.
* Akzeptierzustand ist bei der Schnittmenge dort, wo beide Automaten einen Akzeptierzustand haben.

Andere Akzeptierzustände:

* Vereinigung : Alle Zustände, welche bei mindestens einem Automaten ein Akzeptierzustand sind.
* Differenz : Alle Zustände, welche nur beim ersten Automaten ein Akzeptierzustand sind.

### Mengenoperationen

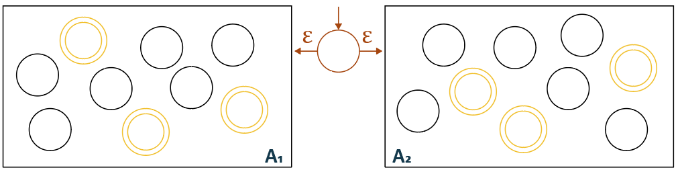
Wenn A1 und A2 DEA’s sind, so kann man Mengenoperationen darauf anwenden und erhält so immer wieder einen DEA, also wieder eine reguläre Sprache.

Operationen:

* Schnittmenge:
* Vereinigung:
* Differenz:

## Reguläre Operationen und reguläre Ausdrücke

### **Reguläre Operationen**

* Vereinigung/Alternative (Oder):  
   Neuer Startzustand mit ℰ-Übergängen zu den einzelnen Automaten. Man kann entweder in den einen oder anderen Automaten gehen (ODER). Muss nicht wissen, was die einzelnen Automaten machen.
* *Ein Bild, das Kreis, Diagramm, Schrift, Design enthält.

  Automatisch generierte Beschreibung*Verkettung:   
  A1 Akzeptierzustände mit ℰ-Übergängen zu Startzugang A2. Aus den vorherigen Akzeptierzuständen von A1 ℰ -Verknüpfung in A2. A1 hat keine Akzeptierzustände mehr.
* Ein Bild, das Kreis, Screenshot, Diagramm, Reihe enthält.

  Automatisch generierte Beschreibung\*-Operation:  
  Neuer Startzugang mit ℰ-Übergang zum alten Startzustand, ℰ-Verknüpfung aus den vorherigen Akzeptierzuständen von A zum neuen Startzugang. Leeres Wort ist ebenso akzeptiert.

### **Reguläre Ausdrücke**

VNEA ist ein NEA, dessen Pfeile mit regulären Ausdrücken angeschrieben sind.  
Zu jedem regulären Ausdruck gibt es einen DEA.

|  |  |  |  |
| --- | --- | --- | --- |
| Ausdruck | Sprache | Bedeutung | NEA |
|  |  | Die leere Sprache |  |
|  |  | Sprache, die nur das leere Wort enthält |  |
|  |  | Sprache, die nur enthält |  |
|  |  | Sprache, die bestimmte Buchstaben enthält |  |
|  |  | Jedes Zeichen aus wird akzeptiert |  |
|  |  | oder Alternative / Vereinigung von zwei Sprachen |  |
|  |  | Verkettung von zwei Sprachen |  |
|  |  | Beliebig viele |  |
|  |  | Mindestens ein plus beliebig viele |  |
|  |  | Das leere Wort oder ein Mal |  |
|  |  | Genau zwei Mal |  |
|  |  | Zwei oder drei Mal |  |
|  |  | Höchstens drei Mal |  |
|  |  | Mindestens drei Mal |  |

### **Umwandlung regex -> NEA**

* Von innen nach aussen übersetzen (Klammern zuerst)
* Einzelne Teile mittel -Übergängen verbinden

Beispiele

|  |  |
| --- | --- |
|  |  |
|  | Ein Bild, das Screenshot, Kreis, Diagramm, Schrift enthält.  Automatisch generierte Beschreibung |

### **Umwandlung VNEA zu regulärem ausdruck**

* Sichergehen, dass vom Startzugang zur Pfeile wegführen und beim Akzeptierzustand nur Pfeile ankommen: Neue Zustände S und E hinzufügen  
  Ein Bild, das Diagramm, Schrift, Muster, Design enthält.

  Automatisch generierte Beschreibung
* Alle Zwischenzustände der Reihe nach entfernen  
  Ein Bild, das Schrift, Diagramm, Screenshot enthält.

  Automatisch generierte Beschreibung
  + 1. Beispiel

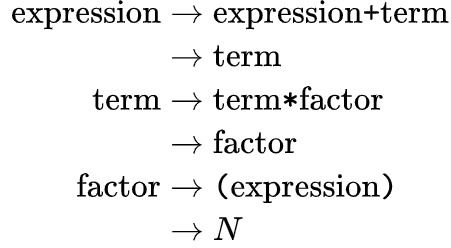
Ein Bild, das Text, Screenshot, Schrift, Kreis enthält.

Automatisch generierte Beschreibung

## Kontextfreie Sprachen

Kontextfreie Sprachen können nur von nichtdeterministischen Stackautomaten erkannt werden. Existiert ein Stackautomat, Regex oder eine Grammatik, ist die Sprache kontextfrei.

Kontextfrei bedeutet, dass bei den Regeln bei der Variable auf der linken Seite der Kontext nicht relevant ist.

* Regeln ohne Kontext:

Beispiel einer Grammatik

* Regeln mit Kontext:

### Kontextfreie Grammatik

Kontextfreie Grammatik:

* : Endliche Menge von Variablen
* : Endliche Menge von Zeichen, Terminalsymbole (Alphabet)
* : Eine Menge von Regeln der Form mit
* : Startvariable
  + 1. Beispiele

**gültige Klammerung**

* + 1. Grammatik für reguläre Operationen

und sind kontextfreie Sprachen mit Grammatiken . Grammatik für reguläre Operationen:

* Neue Startvariable
* Variablen
* Geeignet erweitere Regeln

=>

* Alternative / OR: Regeln für :
* Verkettung / AND: Regeln für
* \*-Operation: Regeln für

### Parse-Tree

**Ein Bild, das Entwurf, Zeichnung enthält.

Automatisch generierte Beschreibung**Zwei Ableitungen eines Wortes einer kontextfreien Sprache heissen äquivalent, wenn sie den gleichen Ableitungsbaum (Parse-Tree) haben, Hat eine Sprache Wörter mit verschiedenen Parse-Trees, heisst sie mehrdeutig.

Beispiel mehrdeutiger Parse Tree

### Chomsky-Normalform (CNF)

Eine Regel ist in der Chomsky-Normalform, wenn auf der rechten Seite nicht vorkommtund jede Regel von der Form oder ist. Zusätzlich ist noch die Regel erlaubt.

* + 1. Umwandlung in Chomsky-Normalform
* Neue Startvariable
* -Regeln: kann weggelassen werden
* Unit-Rules: aus kann man wie aus auch machen
* Verkettungen: ersetzen durch und falls ein Terminalsymbol ist:
  + 1. Beispiel

**Ausgangsgrammatik**

Schritt 1: Neue Startvariable

Schritt 2: nur aus Startvariable, das heisst und sind fakultativ

Schritt 3: Keine Unit-Rules

Schritt 4: Auf rechten Seite genau 2 Variablen oder genau ein Terminalsymbol

(Nicht Variable und Terminalsymbole gemischt)

* + 1. Anwendung der Chomsky-Normalform

Ein Bild, das Text, Diagramm, Reihe, Schrift enthält.

Automatisch generierte BeschreibungChomsky-Normalform ist nicht eindeutig, es kommt auf die Reihenfolge darauf an, wie die Schritte angewendet werden. Man kann sie deshalb nicht brauchen, um Grammatiken zu vergleichen.

* Ableitung eines Wortes ist immer in Regelanwendungen möglich
  + 1. Deterministisches Parsen

Ist aus ableitbar? In Zeichen 

Spezialfälle: und -> ist ein Terminalsymbol. Falls

### CYK-Algorithmus (Cocke-Younger-Kasami)

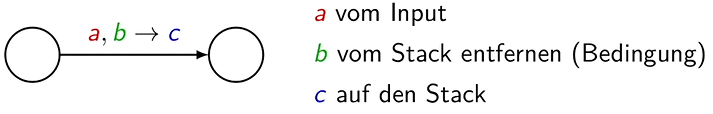
Damit lässt sich der Parse-Tree finden.   
Nachteil: Laufzeit ist

* Zeile 0: Terminalsymbol-Regeln
* Zeile 1: Variablen, die Paare produzieren können
* …

### Stackautomat / Push Down Automat (PDA)

Ein Bild, das Text, Screenshot, Schrift, Zahl enthält.

Automatisch generierte BeschreibungEin Stackautomat kann nur kontextfreie Grammatiken erkennen. Er ist immer nicht deterministisch.

Stackautomat

|  |  |  |
| --- | --- | --- |
| Zeichen | Bezeichnung | Beispiel |
|  | Zustände |  |
|  | Eingabe-Alphabet |  |
|  | Stack-Alphabet |  |
|  |  |  |
|  | Startzustand |  |
|  | Akzeptierzustände |  |

Übergänge: Siehe Bild, gelesen als «Verarbeite Input und ersetze auf dem Stack durch ».

* + 1. Ein Bild, das Diagramm, Kreis, Reihe enthält.

       Automatisch generierte BeschreibungEin Bild, das Screenshot, Text, Stecker enthält.

       Automatisch generierte BeschreibungBeispiel

Jedes wird auf den Stack gelegt. Bei jedem wird wieder eine vom Stack entfernt.

* + 1. Stackautomat standardisieren rote Inhalte ergänzen
* Nur ein Akzeptierzustand: neuer Akzeptierzustand und Übergänge   
  Ein Bild, das Kreis, Symbol, Schrift, Grafiken enthält.

  Automatisch generierte Beschreibung
* Stack leeren: Am Ende soll nichts mehr auf dem Stack sein.  
  Ein Bild, das Kreis, Schrift, Text, Diagramm enthält.

  Automatisch generierte Beschreibung
* Jeder Übergang legt entweder ein Zeichen auf den Stack oder entfernt eines:Ein Bild, das Kreis, Screenshot, Reihe, Diagramm enthält.

  Automatisch generierte Beschreibung
  + 1. Grammatik ablesen

Ausgangspunkt: standardisierte Grammatik mit Starzustand und

* Startvariable:
* Regeln:   
  Ein Bild, das Diagramm, Reihe enthält.

  Automatisch generierte Beschreibung

Beispiel

|  |  |  |  |
| --- | --- | --- | --- |
| Start-Automat | Normalisierung Stackautomat | Grammatik ablesen | Vereinfachen |
|  |  |  |  |

### Ein Bild, das Dreieck, Reihe enthält. Automatisch generierte BeschreibungPumping Lemma für kontextfreie Sprachen

Es gibt auch Sprachen, die nicht kontextfrei sind. Z.B. die Sprache . Um zu beweisen, dass eine Sprache nicht kontextfrei ist, kann man das Pumping Lemma verwenden, muss aber die Grammatik dazu nehmen.

Ist eine Kontextfreie Sprache (CFL), dann gibt es eine Zahl , die Pumping Length, derart, dass jedes Wort mit in fünf   
Teile zerlegt werden kann, dass

Mit dem Pumping Lemma kann man beweisen, dass eine Sprache nicht kontextfrei ist.

(nur v und y gleich viel pumpen, wie bei Sprache mit Klammern () )

Beispiel:

1. Annahme ist kontextfrei

2. Pumping Length

3. Wort (Achtung, und nicht )

4. Unterteilung/Aufteilung (Zerlegung)

Zuerst N aufteilen und dazwischen die Zeichen /   
 Buchstaben schreiben. Danach (uv)xyz einzeichnen  
Ein Bild, das Screenshot, Reihe, parallel, Diagramm enthält.

Automatisch generierte Beschreibung

5. Pumpen

Beim Pumpen nimmt die Anzahl der und zu, nicht aber die Anzahl der

6. Widerspruch ist nicht kontextfrei.

(Im Widerspruch zum Pumping Lemma.

)

## Turing Maschinen Eine Turing Maschine besteht aus:

* Einem unendlich langem Speicherband mit unendlich vielen sequentiell angeordneten Feldern. Pro Feld kann genau ein Zeichen aus einem vordefinierten Alphabet gespeichert werden.
* Einem Lese- und Schreibkopf, der sich auf dem Speicherband feldweise bewegen kann.

Eine Sprache heisst «Turing-Erkennbar», wenn es eine Turing-Maschine gibt, die das Wort akzeptiert.

Ein Bild, das Text, Schrift, Diagramm, Entwurf enthält.

Automatisch generierte BeschreibungTuring Maschine

|  |  |  |
| --- | --- | --- |
| Zeichen | Bezeichnung | Beispiel |
|  | Zustände |  |
|  | Eingabe-Alphabet |  |
|  | Stack-Alphabet |  |
|  |  |  |
|  | Startzustand |  |
|  | Akzeptierzustand |  |
|  | Ablehnungszustand |  |

Zustandsdiagramm

*Ein Bild, das Schrift, Symbol, Logo, Grafiken enthält.

Automatisch generierte Beschreibung*

Übergang

* Übergang möglich, wenn unter dem Schreib-/Lese-Kopf
* Aktuelles Feld auf dem Band wird mit überschrieben
* Kopfbewegung: links, rechts

### Ablauf Programm

* Inputwort auf Band schreiben
* Schreib- / Lesekopf auf erstes Zeichen Positionieren (meist )
* Maschine starten, Einzelschritte ausführen
* Maschine hält in oder an und akzeptiert oder verwirft Wort entsprechend

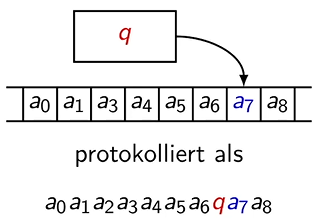
### Beispiele

|  |  |
| --- | --- |
|  | |
| Ein Bild, das Zeichnung, Entwurf, Diagramm, Reihe enthält.  Automatisch generierte Beschreibung | |
| Binärzahlen um 1 erhöhen | Binärzahlen durch 3 Teilbar (DEA in Turing umwandeln) |
|  |  |

### Berechnungsgeschichte

Kann auch auf ein Ausfüllrätsel reduziert werden und somit beweisen, dass SAT NP-Vollständig ist

**Notation für Maschinenzustand**



**Übergang**

Ein Bild, das Schrift, Reihe, Diagramm, Screenshot enthält.

Automatisch generierte Beschreibung

**Protokoll einer Berechnung**

Ein Bild, das Text, Diagramm, Kreis, Design enthält.

Automatisch generierte Beschreibung

### Varianten von Turing-Maschinen

Jede Sprache, die von einer alternativen Turingmaschine erkannt werden kann, kann auch von einer einspurigen Turingmaschine erkannt werden, ist einfach langsamer.

* Anderes Bandalphabet: Simuliert z.B. durch binäre Codierung,
* Mehrere Bänder (ein Pointer pro Band): Simulierbar mit mehrspurigem Band, jedes Band bekommt eine zusätzliche Spur für die Position. Die Position wird über ein Zeichen gelöst, welches verschoben wird und die Position anzeigt.   
  Ein Bild, das Diagramm, Reihe, Schrift, Screenshot enthält.

  Automatisch generierte Beschreibung
* Mehrspurige TM (ein Pointer): Simuliert, in dem man die Spuren alle seriell hinterlegt.   
  Ein Bild, das Diagramm, Zahl, Schrift, Reihe enthält.

  Automatisch generierte Beschreibung
* Nichtdeterministische TM: Bei jedem Übergang maximal verschiedene Möglichkeiten. Mehrere Berechnungswege möglich, Akzeptiert sobald ein Weg zu führt. Simulation: Maximal .   
  3 Bänder: Arbeitsband, Kopie von und Liste aller Folgen von Wahlmöglichkeiten
* Aufzähler: Eine TM mit einem Drucker, bei dem gefundene Wörter «ausgedruckt» werden können. Kann endlos laufen. Wort kann mit «Ausdruck» verglichen werden um akzeptiert zu werden. Aufzählbare Sprache Turing-Erkennbare Sprache

### EntscheidbarkeIt

Ein Entscheider ist eine Turing-Maschine, die auf jedem beliebigen Input anhält. Eine Sprache heisst entscheidbar, wenn es einen Entscheider gibt mit . Man sagt, entscheidet .

* + 1. Turing-erkennbare Sprache

Eine Sprache heisst Turing-erkennbar, wenn es eine TM gibt, die die Sprache als Input akzeptiert. Die TM kann auch unendlich lange laufen, wenn kein akzeptierter Input eingegeben wird. Ausser wenn ein Entscheider ist, dann hält es immer an.

* + 1. Entscheidbare Probleme

Leerheitsproblem M ist eine Turing-Maschine und M hält auf leerem Band, ALL = Gegenteil von Leerheitsproblem  
 Gleichheitsproblem, Akzeptanzproblem, CFG = context free grammar

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Problem | Wort | Bedingung | Entscheidbar | Entscheidungsalgorithmus |
|  | ⟨A⟩ |  | Ja | Minimalautomat hat keinen Akzeptierzustand |
|  | ⟨G⟩ |  | Ja | Chomksy-Normalform |
|  | ⟨M⟩ |  | Nein | Reduktion möglich: |
|  | ⟨A1, A2⟩ |  | Ja | Vergleich der minimalen Automaten, oder Leerheitsproblem bei: |
|  | ⟨G1, G2⟩ |  | Nein | Es gibt eine Reduktion    . Da  ALLCFG  nicht entscheidbar, auch EQCFG nicht entscheidbar. |
|  | ⟨M1, M2⟩ |  | Nein | Weil EQPDA nicht entscheidbar |
|  | ⟨A, w⟩ |  | Ja | Regex-Engines simulieren beliebige DEAs auf beliebigen Input-Wörtern |
|  | ⟨G, w⟩ |  | Ja | CYK, deterministischer Parse-Algorithmus |
|  | ⟨M, w⟩ |  | Nein | Halteproblem |
|  | ⟨M, w⟩ |  | Nein | Auf ATM reduziert -> nicht entscheidbar. |
|  |  |  | Nein | (Gegenteil von E Problem mit TM) |
|  | ⟨M, w⟩ |  | Nein | Reduktionsabbildung der Gegenteil akzeptiert |
|  |  |  |  | wenn Chomsky-Normalf. Regel enthält |

* + 1. Ein Bild, das Diagramm, Zeichnung, Entwurf enthält.

       Automatisch generierte BeschreibungReduktion

Mittels Reduktion kann bewiesen werden, dass eine TM nicht entscheidbar ist. Dazu reduziert man eine TM auf eine nicht entscheidbare TM (. Idee: Neues Problem schaffen, welches allenfalls einfacher zu lösen ist. ( ist leichter entscheidbar als )

Berechenbare Abbildung:

Ist entscheidbar, dann ist auch entscheidbar. Ist nicht entscheidbar, dann ist auch nicht entscheidbar.

|  |  |
| --- | --- |
|  |  |

**Satz von Rice**

Ist eine nichttriviale Eigenschaft Turing-erkennbarer Sprachen, dann ist   
 **hat Eigenschaft** nicht entscheidbar.

Voraussetzung: keine Einschränkung machen, wie Zahlengrösse reduzieren

Definition nichttrivial: Eine Eigenschaft Turing-erkennbarer Sprachen heisst nichttrivial, wenn es zwei Turing-Maschinen und gibt, wobei die Eigenschaft hat und nicht.

Folgerung: Es ist nicht möglich, einem Programm anzusehen, ob die akzeptierte Sprache eine nichttriviale Eigenschaft hat.

Lösungsanleitung einer Prüfungsfrage:

- Nichttriviale Eigenschaft aufschreiben

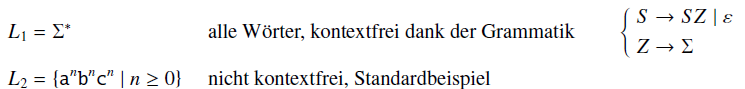
- Die beiden Sprachen und bilden (*meistens kann man die leere Menge oder als eine dieser Sprachen verwenden)*

- Gibt es ein Programm, welches beide Sprachen erkennen kann? Sind beide Sprachen Turing erkennbar?

- Dann besagt der Satz von Rice, dass die Sprache nicht entscheidbar ist.

Beispiel 1

Zeigen Sie mit Hilfe des Satzes von Rice, dass die Frage, ob eine Turing-Maschine eine kontextfreie Sprache akzeptiert, nicht entscheidbar ist.   
  
Ob eine Sprache kontextfrei ist oder nicht ist eine nichttriviale Eigenschaft. Um dies zu zeigen, muss man zwei Sprachen angeben, von denen die eine Kontextfrei ist und die andere nicht. Zwei mögliche Sprachen sind



Da die Spracheigenschaft, kontextfrei zu sein, eine nichttriviale Eigenschaft ist, folgt nach dem Satz von Rice, dass nicht entscheidbar ist, ob eine Turing-Maschine eine kontextfreie Sprache akzeptiert.

Beispiel 2  
Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte Beschreibung

## Komplexität

Eine Turing-Maschine mit mehreren Bändern und Laufzeit kann in Laufzeit auf einer Standard Turing-Maschine simuliert werden.

Sei aber eine Nichtdeterministische Turing-Maschine, dann kann diese in Laufzeit simuliert werden. Dies gilt nur für NTM, die auch Entscheider sind.

### Polynomielle und Exponentielle Laufzeit

Ein Bild, das Text, Reihe, Schrift, Screenshot enthält.

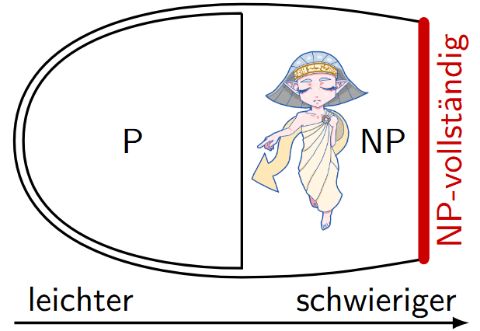
Automatisch generierte Beschreibung**Polynomielle Probleme** Skalierbarkeit

* Gauss-Alogrithmus
* FFT
* Sortieren

**Exponentielle Probleme** Sicherheit

* Simplex-Algorithmus für lineare Optimierung
* Ganzzahlige Optimierung
* Faktorisierung von grossen Zahlen (RSA)
* Diskreter Logarithmus (Diffie-Hellmann)
* Hash-Kollision

### Klasse P und NP

Die Klasse P besteht aus den Sprachen, die mit einem Entscheider (DTM) in polynomieller Laufzeit entschieden werden können. Ist eine Teilmenge von NP.

Die Klasse NP besteht aus Sprachen, die mit einer nichtdeterministischen Turingmaschine (NTM) in polynomieller Laufzeit entschieden werden können. Eine Sprache ist genau dann in NP, wenn sie in polynomieller Zeit verifiziert werden kann.

Nicht verwechseln mit NP-vollständig: Eine entscheidbare Sprache B heisst **NP-vollständig**, wenn sich jede Sprache A in NP polynomiell auf B reduzieren lässt:  NP-vollständige Probleme sind alle gleich schwierig. Falls P /= NP, dann können NP-vollständige Probleme nicht in polynomieller Zeit gelöst werden.

* + 1. Polynomieller Verifizierer (Gleichbedeutend mit NP)

Ein Bild, das Text, Screenshot, Reihe, Schrift enthält.

Automatisch generierte BeschreibungEin polynomieller Verifizierer ist eine Turingmaschine, die die «Lösung» überprüfen kann.   
Für die Sprache ist das eine Turingmaschine , so dass es für jedes ein Wort c (das Lösungszertifikat) gibt, für das gilt: akzeptiert

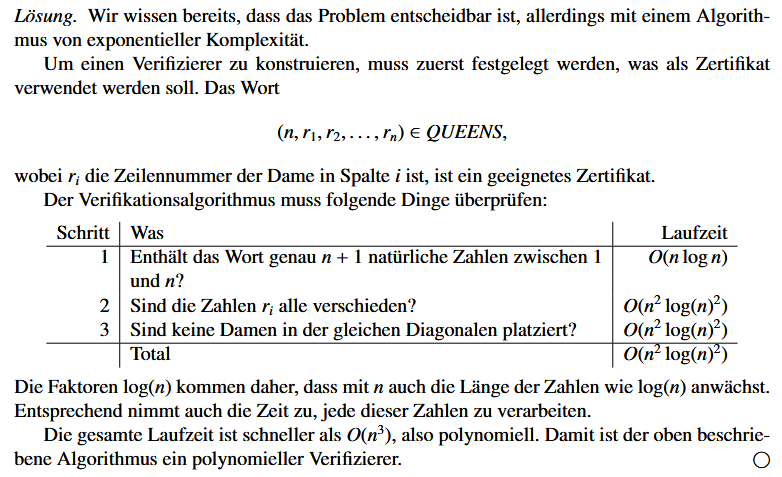
Die Laufzeit von ist polynomiell in .

**Zertifikat** ist wie ein Wunschzettel, wo man alles angeben kann, dass einem bei der Verifikation hilft.

**Wichtig:** Entscheidbar und Zertifikat beantworten mit Begründung! Entscheidbar meistens: Alle Möglichkeiten durchprobieren, wieviele?

Beim Aufstellen des Verifizierers immer auch prüfen (und begründen), ob das Problem überhaupt entscheidbar ist. Die Tatsache, dass es ein Zertifikat gibt, reicht dazu nicht aus. Man muss spezifizieren, was als Zertifikat angefordert wird. Am Ende muss eine Antwort auf die gestellte Frage stehen. Es reicht nicht, einfach nur die Schritte hinzuschreiben.

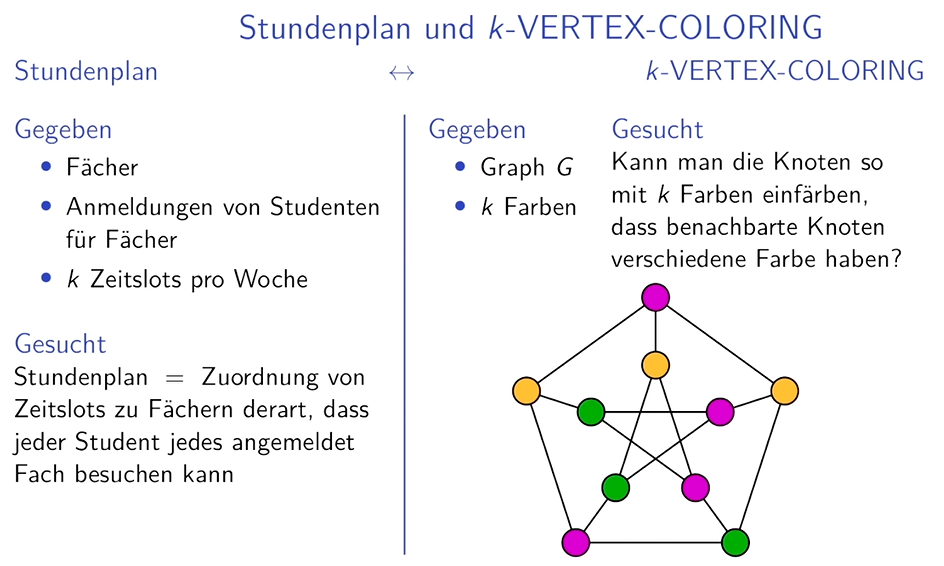
Hier geht es darum, ob eine Sprach in NP ist. Das Wort «NP-Vollständig» hat hier nichts verloren!

Beispiele

|  |  |
| --- | --- |
| Sudoku | Nurikabe  *c= Liste der schwarzen Felder* |
|  |  |

* + 1. Reduktion

Dient dazu, zwei Probleme zu vergleichen.

* Fach Knoten
* Anmeldung Kante
* Zeitslot Farbe

Polynomielle Reduktion

Ein Bild, das Text, Screenshot, Schrift, Zahl enthält.

Automatisch generierte Beschreibung

### NP-Vollständig

Eine Entscheidbare Sprache heisst NP-Vollständig, wenn sich jede Sprache in NP polynomiell auf reduzieren lässt. Sind die schwersten Probleme in NP Katalog von Karp

Es genügt nicht, das Vergleichsproblem zu erkennen, es muss auch eine 1-1-Reduktion durchgeführt werden. Und am Ende muss eine Antwort auf die Frage stehen. Und irgendwo muss das Wort NP-vollständig stehen, NP reicht nicht!

## Katalog von Karp

Probleme mit K Zahlen

CLIQUE-COVER

VERTEX-COVER

SET-PACKING

SET-COVERING

EXACT-COVER

K-CLIQUE

FEEDBACK-\*-SET

VERTEX-COLORING

SEQUENCING

STEINER-TREE

(MAX-CUT)

Aufteilung in zwei Teilmengen

PARTITION

MAX-CUT

Probleme mit Zahl 3

3D-MATCHING

3SAT

Unterschiede Hitting-Set, Exact-Cover

HITTING-SET: Menge von Punkten

EXACT-COVER: Menge von Teilmengen EXACT-\*-COVER: disjunkt?

Unterschiede Feedback-\*-Set

NODE: Vertex entfernen

ARC: Kanten entfernen

Unterschiede Set-\*

Covering: Endliche Familie endlicher Mengen

Packing: Eine Familie von Mengen

|  |  |  |  |
| --- | --- | --- | --- |
|  | SET-COVERING | SET-PACKING | EXACT-COVER |
| Anzahl Mengen | k | k | (frei) |
| Vereinigung | ganze Menge |  | ganze Menge |
| paarweise Schnittmengen |  | ∅ | ∅ |

Unterschiede \*-Cover

Graph oder Mengen mit einer Zahl k

CLIQUE: k-Cliquen vorhanden?

VERTEX: mit k Knoten möglich?

Weitere (Karp-Baum unten)

HAMCIRCUIT & UHAMCIRCUIT, SAT, BIP, SUBSET SUM

SAT(Logik), 3SAT (Logik)

Gibt es eine mögliche Lösung der **Booleschen Gleichung**, damit sie TRUE ergibt.  
3SAT: Klausel besteht höchstens aus 3 Literalen pro Klausel und ist in KNF

Klausel = Klammerausdruck, Literale = Variable

Bsp. Elektriker Variablen: n-Schalter -> n-Variablen,m-Räume -> m-Klauseln**,** Überall Licht -> Aussage = true**,**Stromkreise -> Wertebereich (True, False)

Man kann Wahrheitstabellen bilden. NP-Vollständig ist jedes Problem. Das sich auf SAT reduzieren lässt.

SUBSET-SUM (RUCKSACK-PROBLEM)

Gegeben: Menge S von ganzen Zahlen. Kann man eine Teilmenge finden, die als Summe einen bestimmten Wert t hat? (S ist eigentlich keine Menge, denn gleiche Zahlen können mehrmals vorkommen.)

Beispiel: Rucksack-Problem

Variablen: Grösse der Gegenstände -> Zahl, Menge der Gegenstände -> Menge S der Zahlen, Grösse des Rucksacks -> Wert t (Summe), Gegenstände im Rucksack -> Teilmenge

Ein Bild, das Reihe, Kinderkunst, Zeichnung enthält.

Automatisch generierte Beschreibung***k-CLIQUE***

Gibt es k Knoten, die alle miteinander verbunden sind?

k: Anzahl Vertizes in einere Clique (gegeben)

Gibt es eine solche Clique in dem Graphen (Siehe Bild)?

für ja (eingezeichnet)

für nein

**Beispiel:** Job-Parallelisierbarkeit: gegeben eine Menge von n Jobs, die jeweils exklusiv auf m Ressourcen zugreifen, Jobs dürfen nicht gleichzeitig laufen. Zeigen Sie, dass das Problem zu entscheiden, ob mit diesen Jobs zu irgendeinem Zeitpunkt mehr als k Prozessoren ausgelastet werden können, NP-vollständig ist.  
**Variablen:** Graph (Job-Parallelisierbarkeit), Knoten (n Job), Kante (m gleiche Ressourcen), k-Clique (Auswahl Knoten ohne Verbindung), k (Auslastbare Anzahl Prozessoren)

*Ein Bild, das Kinderkunst, Zeichnung, Entwurf, Reihe enthält.

Automatisch generierte Beschreibung****CLIQUE-COVER (EXOR mit bestimmter Zahl k)***

Gegeben: Ein Graph G und eine **positive Zahl k**. Gibt es k Cliquen so, dass jede Ecke in  
 genau einer der Cliquen ist?

***Beispiel:*** *Für eine Gruppenarbeit sollen k Gruppen gebildet werden. Um die Zeit für das Kennenlernen   
möglichst kurz zu halten, sollen sich die Leute einer Gruppe bereitS kennen. Alle Leute sollen eingeteilt sein.*

***Variablen:*** *Teilnehmer -> Knoten, Kennen sich -> Kante, Anzahl Gruppen -> k, Gruppe -> Clique*

*Ein Bild, das Kinderkunst, Reihe, Zeichnung, Entwurf enthält.

Automatisch generierte Beschreibung****VERTEX-COVER***

Gegeben: Ein Graph G und eine Zahl k. Gibt es eine **(Teil)menge** W von k Knoten so, dass   
jede **Kante** von G einen Endpunkt in dieser Teilmenge W hat?

***Beispiel:*** *Gibt es 4 Knoten so, dass jeder andere Knoten eine Kante zu diesem hat?*

***Beispiel:*** *Ein Verkehrsnetz soll regelmässig durch Mitarbeiter kontrolliert werden, die ihre Basis an   
einzelnen Knotenpunkten des Netzes haben, Kann man auf effiziente Art herausfinden, an welchen   
Knotenpunkten man Kontrolleure stationieren muss, damit jede Strecke in einem Knoten mit   
Kontrolleur endet?****Variablen:*** *Knotenpunkte -> Knoten, Strecke -> Kante, Anzahl Kontrolleure (möglichst wenig) -> k,   
Knoten mit Kontrolleur -> Knoten aus dem Vertex-Cover*

***SET-PACKING***

Gegeben: eine Familie und eine Zahl

*Wie viele der Mengen Si kann man in U hineinpacken, ohne dass sie sich überlappen?* Gibt es eine k-elementige Teilfamilie mit , d. von paarweise disjunkten Teilmengen?

Menge S und Menge T von Teilmengen, Gibt es k Teilmengen in T, welche disjunkt sind und die Menge S abdecken?

*Ein Bild, das Zeichnung, Entwurf, Lineart, Clipart enthält.

Automatisch generierte Beschreibung*Beispiel 1 Küche: *In einer Küche hat man eine Menge an Zutaten und ein  
Rezeptbuch voller Rezepte. Nun möchte man möglichst viele der Rezepte kochen,   
ohne eine Zutat mehrmals zu verwenden.*

Beispiel 2 Medizinstudie: *Für eine medizinische Studie ist eine grosse Zahl von   
Probanden rekrutiert worden. Sie sind bereits auf Allergien getestet worden,   
man weiss also von jedem Probanden, auf welche Allergene er allergisch reagiert.   
Die Untersuchung soll sich auf eine Teilmenge von oder noch mehr   
ausgewählten Allergenen beschränken, die so beschaffen ist ,dass kein Proband   
auf mehr als eines der ausgewählten Allergene reagiert.*

***Variablen:*** *I (Allergene), Si (Auf Allergen i allergische Probanden), Ausschlussbedinungn zwischen Allergenen i und j, Teilmenge*

Ein Bild, das Diagramm, Reihe, Kreis enthält.

Automatisch generierte Beschreibung***SET-COVERING***

Gegeben: Eine **endliche Familie endlicher Mengen** und eine   
**Zahl k**. (Familie = Set von Sets aber Set doppelt erlaubt)  
*Gibt es eine Unterfamilie bestehend aus k Mengen, die die gleiche Vereinigung U hat?*   
Kann man k Teilmengen bilden, welche die Menge S komplett abdecken?

***Beispiel:*** *Von jedem Aussichtspunkt aus kann man eine endliche Anzahl Sehenswürdigkeiten erkennen. Ein Tourist, der es sehr eilig hat, möchte mit dem Besuch von nur k Aussichtspunkten alles sehen, was man an Sehenswürdigkeiten von allen Aussichtspunkten aus sehen könnte. Warum ist es schwierig, für den Touristen eine Auswahl zu treffen?*

***Variablen:*** *Aussichtspunkt ↔ Sj, Sichtbare Sehenswürdigkeiten ↔ , Auswahl von Aussichtspunkten ↔ ,   
Bedingung kann alles sehen↔ S=*

EXACT-COVER

Gegeben: Eine Familie von Teilmengen einer Menge U. Gibt es eine Unterfamilie von Mengen, die disjunkt sind, aber die gleiche Vereinigung haben? Jedes Element in U soll genau in einer der Teilmengen der Familie S vorkommen. Die gesuchte Menge bildet eine exakte Überdeckung.

Ein Bild, das Kreis enthält.

Automatisch generierte BeschreibungBeispiel: Student Xaver Tecco soll im Rahmen einer Big-Data-Studienarbeit die Kunden einer grossen Shop-Website untersuchen und klassifizieren. Es steht eine grosse Zahl von binären Eigenschaften zur Verfügung,   
zum Beispiel ob Kunden ein bestimmtes Produkt gekauft haben, oder ob ein Kunde nur  
im Dezember einkauft. Herr Tecco soll herausfinden, ob es eine Teilmenge von Kriterien   
gibt, dass jeder Kunde genau eine der Eigenschaften hat.

Variablen: Eigenschaft -> Menge , Teilmenge von Eigenschaften -> Unterfamilie ,  
Genau eine der Eigenschaften, Alle Kunden erfasst

*Ein Bild, das Reihe, Kreis enthält.

Automatisch generierte Beschreibung****VERTEX-COLORING (Planungsproblem)***

Kann man die Knoten so mit k Farben einfärben, dass benachbarte Knoten   
verschiedene Farben haben?

***Beispiel:*** *Job-Planung*

***Variablen:*** *n Job -> Knoten, m gleiche Ressourcen -> Verbindungen, parallele Jobs (bzw. Intervall)   
-> gleiche Farbe, N Intervalle -> n Farben*

3D-MATCHING (Heiratsproblem+Wohnungen, ähnlich wie EXACT-COVER)

Gegeben: Endliche Menge T und Menge U von Tripeln . Gibt es eine Teilmenge von so, dass und keine zwei Elemente von w stimmen in irgendeiner Koordinate überein?

Ein Bild, das Diagramm, Reihe, Plan, technische Zeichnung enthält.

Automatisch generierte BeschreibungBeispiel: Ein Koch hat je n Rezepte für Vorspeisen, Hauptspeisen und Desserts. Nicht alle Vorspeisen lassen sich mit jeder Hauptspeise kombinieren, dasselbe gilt auch für Desserts. Damit jeder   
seiner Rezepte regelmässig zum Einsatz kommt, möchte der Koch   
eine Folge von n Menus zusammenstellen, sodass jedes   
Rezept in genau einem der Menus vorkommt.

Variablen: Nummern der Rezepte -> Menge T,

Menuzusammenstellungen -> Tripel aus ,

Menge der möglichen Menuzusammenstellungen -> Menge U,

Gesuchte Menus n -> Teilmenge W, so

*Ein Bild, das Kinderkunst, Zeichnung, Kreis, Entwurf enthält.

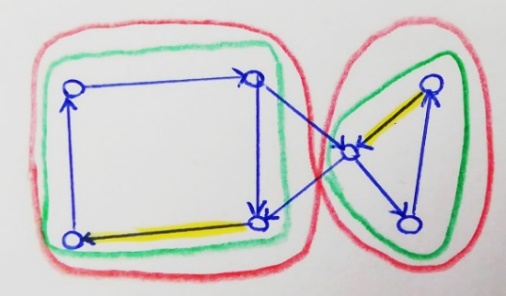
Automatisch generierte Beschreibung****FEEDBACK-NODE-SET***

Gegeben: Ein **gerichteter Graph G** und eine **Zahl k**. Gibt es eine   
endliche Teilmenge von k Vertizes/Knoten von G, sodass jeder   
Zyklus in G **einen Vertex** in der Teilmenge enthält?

***Beispiel:*** *Es gibt mehrere Buslinien. Wo muss das Putzpersonal platziert werden,   
damit alle Linien geputzt werden können? Man möchte möglichst wenig   
Personal einsetzen.*

***Bild:*** *, Alle 3 Umrahmungen = Zyklen. Knoten markieren*

***Variablen:*** *Graph (Plan des Einkaufszentrums), Knoten (Kreuzungsstellen), Kanten (Gänge), Richtung, Anzahl k (Budget für Stationen), Arc set (Platzierungen der Desinfektionsstationen), Zyklus muss definiert sein*

*****FEEDBACK-ARC-SET***

Gegeben: Ein **gerichteter Graph G** und eine **Zahl k**. Gibt es eine   
endliche Teilmenge von k Kanten von G, sodass jeder   
Zyklus in G **eine Kante** aus der Teilmenge enthält?

***Beispiel:*** *Es gibt mehrere Buslinien. Das Personal putzt während der Fahrt.  
Wo muss das Putzpersonal platziert werden, damit alle Linien geputzt werden   
können? Man möchte möglichst wenig Personal einsetzen.*

***Bild:*** *, Kanten markieren****Variablen:*** *Graph (Plan des Einkaufszentrums), Knoten (Kreuzungsstellen), Kanten (Gänge), Richtung, Anzahl k (Budget für Stationen), Arc set (Platzierungen der Desinfektionsstationen)*

*Ein Bild, das Handschrift, Schrift, Reihe, Design enthält.

Automatisch generierte Beschreibung*HAMCIRCUIT & UHAMCIRCUIT (HAMPATH & UHAMPATH)

Ein Hamilton-Pfad in einem gerichteten Graphen ist ein Pfad, der jeden Knoten genau einmal   
enthält (Haus des Nikolaus). Komme zu jedem Knoten genau einmal.

Ungerichtet ist es der Uhamcircuit.   
Beispiel: Ganze Schweiz bereisen ohne eine Stadt mehrmals zu besuchen.

Variablen: Knoten (Block), gerichtete Kante (Fahrt), Hamiltonischer Pfad

BIP

Zu einer ganzzahligen Matrix C und einem ganzzahligen Vektor d, ist ein binärer Vektor x zu finden mit

SUBSET-SUM ≤P BIP

***SEQUENCING***

Gegeben: Ein **Vektor** von Laufzeiten von p **Jobs**, ein Vektor von spätesten **Ausführzeiten**, ein **Strafenvektor** und eine **positive Zahl k**.

Gibt es eine Permutation der Zahlen 1, …, p sodass die Gesamtstrafe für verspätete Ausführung bei der Ausführung der Jobs nicht grösser ist als k? Gegeben: Eine Menge an Jobs, pro Job eine Ausführzeit, Deadline und eine Strafe sowie eine maximale Strafe (k). Die Jobs müssen sequenziell abgearbeitet werden. Wird ein Job zu spät fertig, muss eine Strafe gezahlt werden. Gesucht: Eine Reihenfolge von Jobs so, dass die Strafe kleiner gleich k ist.

***Beispiel:*** *Eine Firma hat eine bestimmte Anzahl laufende Verträge. Der Firma ist es nicht möglich, alle Verträge in einer bestimmten Zeit abzuarbeiten. Sie versucht also Schadensbegrenzung zu machen, indem sie möglichst viele Verträge in der verbleibenden Zeit abarbeitet, die eine hohe Strafe zur Folge haben.*

***Variablen:*** *Ausführungszeit von Job, Deadline, Kosten, Permutation / Reihenfolge*

***PARTITION***

Gegeben: Eine **Folge von S ganzen Zahlen**

Kann man die Indizes 1,2,…,S in zwei Teilmengen und teilen, sodass die Summe der zugehörigen Zahlen identisch ist? Gibt es **zwei disjunktive Teilmengen** mit der **gleichen Summe**?

***Beispiel:*** *Eine Reihe von Wassergläsern ist unterschiedlich gefüllt. Es sollen 2 Behälter gleich voll mit den Gläsern gefüllt werden. Welche Gläser müssen in welche Behälter geleert werden?*

***MAX-CUT***

Gegeben ein Graph G mit einer **Gewichtsfunktion** und eine Zahl W

Gibt es eine **Teilmenge** S der Vertizes, sodass das Gesamtgewicht der Kanten, die S mit seinem Komplement verbinden, mindestens so gross ist wie W?

Ein Bild, das Kinderkunst, Reihe, Diagramm, Kreis enthält.

Automatisch generierte BeschreibungDer Max-Cut eines Graphen ist eine Zerlegung seiner Knotenmenge in zwei Teilmengen, sodass das Gesamtgewicht der zwischen den beiden Teilen verlaufenden Kanten mindestens W wird.

***Beispiel:*** *Feindliche Übernahme einer Firma, mit resultierender Aufteilung der Abteilung,   
dass diese möglichst ineffizient miteinander kommunizieren können.*

***Variablen:*** *Abteilung -> Vertex, Kommunikationsbeziehung -> Kante, Kommunikationsvolumen ->   
Gewicht einer Kante*

HITTING-SET

Gegeben: Eine Menge von Teilmengen . Gibt es eine Menge H, die jede Menge in genau einem Punkt trifft, also   
Gibt es eine Menge W ⊂ U derart, dass W mit jeder Menge Si nur ein Element gemeinsam hat?

*HITTING-SET sucht zu einer Familie von Mengen, eine Menge von Punkten (Elemente der Vereinigung dieser Mengen), die jede Menge der Familie in genau einem Punkt treffen. Man beachte, dass es keine Forderung über die Anzahl der Punkte gibt.*

Gegeben: Gesucht:

Variablen: Punkt in S, Teilmenge Ui aller S, Hitting Set H

*Ein Bild, das Handschrift, Reihe, Diagramm, Design enthält.

Automatisch generierte Beschreibung****STEINER-TREE***

Gegeben: Ein Graph G, eine **Teilmenge R von Knoten**, eine **Gewichtsfunktion** und eine   
**positive Zahl k**.

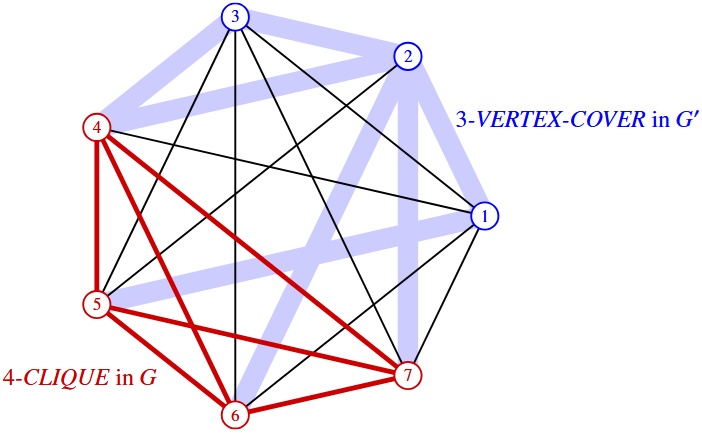
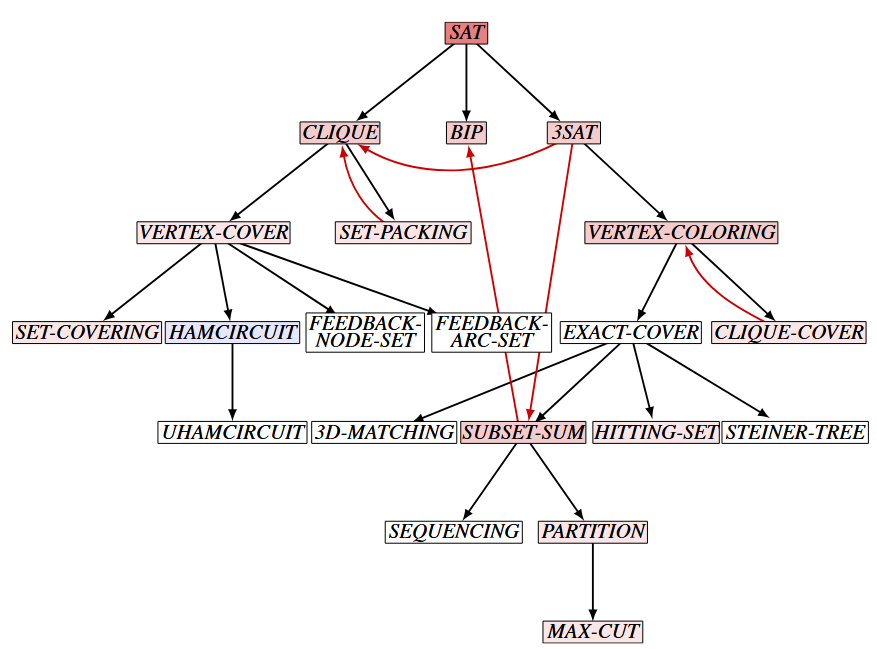
Gibt es einen Baum mit Gewicht , dessen Knoten in R enthalten sind? Das Gewicht des   
Baumes ist die Summe der Gewichte über alle Kanten im Baum.

***Beispiel:*** *Bau einer Zugstrecke oder Stromnetzes:*

***Variablen:*** *Stromnetz / Zugstrecke -> Steiner-Tree, Ortschaften -> Knoten, zu erschliessende Ortschaften  
 -> Knoten in R, Baukosten -> Gewicht w einer Kante, Budget -> maximales Gewicht k*

### **Vorgehen bei der Prüfung**

* Ein Bild, das Diagramm, Plan, technische Zeichnung, Entwurf enthält.

  Automatisch generierte BeschreibungProblem aus Liste suchen, dabei Punkte beachten wie: Ist es ein Mengenproblem, Planungsproblem, kommen k-Zahlen vor, Teilmengen?
* Reduktion des Problems auf Karp mit Pfeil-Liste darstellen (Eigenschaften)
* Zusammenfassung schreiben: «Problem x ist äquivalent mit Karp-Katalog-Problem y und deshalb gehört es zu den NP-Vollständigen Problemen. Daher ist es nicht einfach entscheidbar (Es gibt derzeit keinen polynomiellen Algorithmus).»

## Turing-Vollständigkeit

Turing-Maschinen und moderne Computer haben einige Gemeinsamkeiten. Ein Computer erfüllt alle Eigenschaften einer Turing-Maschine (Zustände, Band, Schreib-/Lesekopf). Einzig kann er mehrere Programme ausführen und eine Turing-Maschine ist problemspezifisch. Es gibt aber die Universelle Turing-Maschine mit

* Eigenem Band für Codierung der Übergangsfunktionen
* Eigenem Band für aktuellen Zustand
* Arbeitsband

Eine solche Mehrbandmaschine kann auch auf einer Standard-Turingmaschine simuliert werden.  
Es gibt aber Komponenten, die ein PC hat, eine Turing-Maschine aber nicht.

* Persistenter Speicher: Files nicht unterscheidbar von Daten, für TM egal
* Interaktion: Lösung eines spezifizierten Problems braucht keine Interaktion
* Input / Output: Output ist nicht wesentlich. Input «existiert nicht». Wenn TM angehalten wird, kann Kernel Band «manipulieren» und TM sieht nun geänderter Input, weiss aber nicht, ob er den Input selber geändert hat oder die Veränderung von aussen kommt.

### **Vergleich von Maschinen**

Eine TM ist «leistungsfähiger» als eine TM , wenn die Maschine simulieren kann.  
 ist simulierbar auf

### Programmiersprachen

Eine Programmiersprache ist Turing-Vollständig, wenn sie folgende Eigenschaften erfüllt:

* Es lässt sich ein Infinite Loop erstellen (auch bei Rekur
* Sie kann auf einen «unendlich» grossen Speicher zugreifen
* Bedingte Ausführung ist möglich

## Glossar

* Die Sprache heisst regulär, wenn es einen DEA A gibt, mit .   
  Beispiel:
* Die Sprache heisst nicht regulär, wenn es keinen DEA A gibt, mit .   
  Beispiel:
* Die Sprache heisst kontextfrei, wenn die Sprache nur von einem nichtdeterministischen Stackautomaten erkannt werden kann

Beispiel:

* Die Sprache heisst nicht kontextfrei, wenn das Pumping Lemma für Kontextfreie Sprachen einen Widerspruchbeweis liefert.  
  Beispiel:
* Eine Sprache heisst «Turing-Erkennbar», wenn es eine Turing-Maschine gibt, die das Wort akzeptiert.  
  Beispiel:

## Anhang / alte Prüfungsfragen

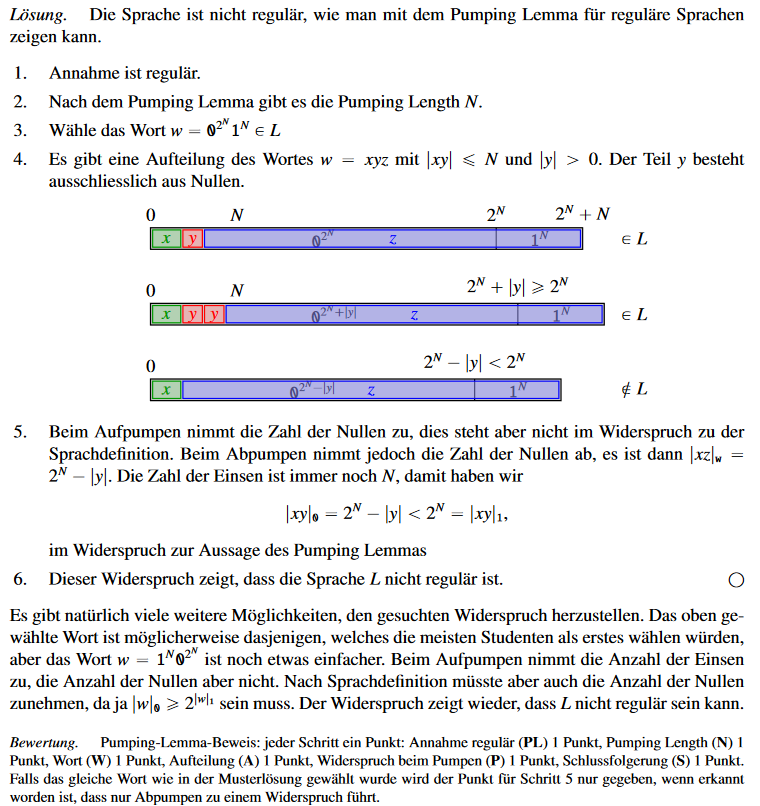
### DEA

Ein Bild, das Text, Screenshot, Diagramm, Reihe enthält.

Automatisch generierte Beschreibung

Wenn gleich statt ungleich, einfach Akzeptier- und nichtakzeptierzustände umkehren

### **Pumping Lemma**



### Verifizierer

Ein Bild, das Text, Screenshot, Schrift, Dokument enthält.

Automatisch generierte BeschreibungEin Bild, das Diagramm, Reihe, Screenshot enthält.

Automatisch generierte BeschreibungEin Bild, das Text, Screenshot, Schrift, Algebra enthält.

Automatisch generierte Beschreibung

### Satz von Rice

In vielen Anwendungen wird verlangt, dass der Output sortiert wird. Es wäre daher nützlich für die Qualitätssicherung, wenn man ein Tool schreiben könnte, welches von einem Programm entscheiden kann, ob sein Output korrekt sortiert wird. Viele Programmiersprachen haben zum Sortieren Funktionen oder Klassen, die Daten sortieren können, man könnte testen, ob diese Klassen vom Code verwendet werden. Manchmal fallen die Daten aber auch automatisch sortiert an, zum Beispiel wenn in einer Datenbank ein Index verwendet wird. Es reicht also nicht, nur die Verwendung der genannten Klassen zu prüfen. Ist es möglich, so ein Tool zu schreiben?

Nehmen Sie der Einfachheit halber an, dass das Tool nur auf Programme angewendet werden soll, welche Wörter aus Zeichen in erzeugen, und die Zeichen in einem Wort sollen alphabetisch aufsteigend sortiert sein.

Ein Bild, das Text, Screenshot, Schrift, Reihe enthält.

Automatisch generierte Beschreibung

### Chomsky-Normalform

Ein Bild, das Text, Screenshot, Schrift, Dokument enthält.

Automatisch generierte Beschreibung

### Ein Bild, das Zeichnung, Diagramm, Entwurf, Reihe enthält. Automatisch generierte BeschreibungZustandsdiagramm analysieren

Betrachten Sie die Turing-Maschine M mit dem Zustandsdiagramm rechts.

* **Wie wird das Wort 001011 verarbeitet?**Die Berechnungsgeschichte für das Wort ist

Ein Bild, das Text, Screenshot, Zahl, Schrift enthält.

Automatisch generierte Beschreibung

* **Welche der Wörter 01, 01000, 01000000, 111, 0111, 000111, 000111111, 00010101111 werden akzeptiert?**  
  Ein Bild, das Text, Screenshot, Schrift, Informationen enthält.

  Automatisch generierte Beschreibung
* **Welche Sprache wird von M akzeptiert?**  
  Ein Bild, das Text, Schrift, Screenshot, Algebra enthält.

  Automatisch generierte Beschreibung

Skizzieren Sie einen Algorithmus ….

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte Beschreibung

Ein Bild, das Text, Screenshot, Schrift, Dokument enthält.

Automatisch generierte Beschreibung

### Pumping Lemma 2

Ein Bild, das Text, Screenshot, Schrift, Dokument enthält.

Automatisch generierte Beschreibung

Ein Bild, das Text, Screenshot, Schrift, Reihe enthält.

Automatisch generierte Beschreibung

Ein Bild, das Text, Screenshot, Schrift, Informationen enthält.

Automatisch generierte Beschreibung



Ein Bild, das Text, Screenshot, Reihe, Diagramm enthält.

Automatisch generierte Beschreibung