# Grundlagen

## Mathe

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| A | B | ¬A∧ | A∧B | A∨B | A → B∧ | A ⟷ B∧ |
| Negation | Konjunktion | Disjunktion | Implikation | Äquivalenz |
| W | W | F | W | W | W | W |
| W | F | F | F | W | F | F |
| F | W | W | F | W | W | F |
| F | F | W | F | F | W | W |

**Bindungsstärke:** ¬ vor∧, ∨vor →, ⟷

**Quantoren:** ∀x = Für alle x, ∃x = Es gibt ein x

Graphical user interface, text, application

Description automatically generated

|  |  |
| --- | --- |
| ≡ | Zu sich selber äquivalent |
| **x** | Akzeptier- und Nichtakzeptierzustand |
| **x** | Übergang führt in ein **x** |
| ≡ | Restliche Paare |

## Begriffe

**Alphabet:** Eine nichtleere endliche Menge . Elemente von heissen Zeichen

Diagram

Description automatically generated**Wort:** Zeichenketter der Länge ist ein -Tupel in . Element von heisst Wort der Länge

**Leeres Wort:** Zeichenkette  der Länge 0 heisst leeres Wort

**Alle Worte:** Menge aller Wörter ist

Wörter haben immer eine endliche Menge

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

**Sprache:** Eine Teilmenge heisst Sprache

**Leere Sprache:** Es gibt 2 Sprachen mit leerem Alphabet

**Wahrheit:** Mit der objektiven Realität übereinstimmende Aussagen

**Konsequenz:** Ohne Beobachtung der Realität keine Wahrheit. Subjektive Erfahrung können nicht als wahr bezeichnet werden

**Annahme:** Objektive Realität, über die sich wahre Gesetzmässigkeiten formulieren lassen

**Falsifizierbarkeit:** Aussage ist falsifizierbar, wenn man mit einer Beobachtung diese als falsch erkannt werden kann

**Beweis:** In der Mathematik als fehlerfrei anerkannte Herleitung der Un/Richtigkeit einer Aussage. Beweis erklärt, warum eine Aussage wahr ist

# Dea

## definition

Übergange zwischen Zuständen in einem DEA sind immer eindeutig.

Ein DEA ist ein Quintupel mit

|  |  |  |
| --- | --- | --- |
|  | 0 | 1 |
| q0 | q2 | q1 |
| q1 /F | q0 | q1 |
| q2 /F | q1 | q2 |

1. Q : Eine beliebige endliche Menge an Zuständen
2. : Eine endliche Menge, genannt das Alphabet
3. : heisst Übergangsfunktion
4. : Startzustand
5. F : Menge der Akzeptierzustände

## myhill-nerode automat

Mit dem Satz von Myhill-Nerode kann die Regularität einer Sprache L erkannt werden. Zu einer beliebigen regulären Sprache kann ein endlicher Automat konstruiert werden.

Diagram

Description automatically generated**Beispiel:**  (Anzahl a im Wort w ist gerade)

|  |  |  |
| --- | --- | --- |
| w | L(w) | L |
| ε | L | L |
| a | L(a) = {w ∈ ∑\* | |w|a ungerade} | L’ |
| b | L(b) = {w ∈ ∑\* | |w|a gerade} = L | L |
| |w|a ungerade | L(a) | L’ |
| |w|a gerade | L | L |

Diese Sprache ist regulär da es eine endliche Menge an Zuständen vorhanden sind.

**Beispiel:**  (Gleich viele 0 wie 1)

|  |  |
| --- | --- |
| w | L(w) |
| ε | L |
| 1 | ∅ |
| 1n | ∅ |
| 0 | {0n1n+1 | n ∈ ℕ} |
| 02 | {0n1n+2 | n ∈ ℕ} |
| 0k | {0n1n+k | n ∈ ℕ} |
| 0k1l | {1k-l } falls k ≥ l |
| 0k1l | ∅ falls k < l |

Ab Zeile 4 sieht man, dass es unendlich viele verschiedene Mengen , somit kann L nicht von einem endlichen Automaten erkannt werden.

## minimaler automat

Zwei endliche Automaten können verschieden aussehen, jedoch die gleiche Sprache akzeptieren. Mit dem minimalen Automat können Automaten auf ihre Standardform gebracht werden und falls diese übereinstimmen, akzeptieren sie die gleiche Sprache.

1. Tabelle aller Zustände erstellen
2. Alle Zustände zu sich selber äquivalent markieren (≡)
3. Alle Zustände zu allen Akzeptierzuständen als nicht äquivalent markieren (x)

→ Ein Akzeptierzustand kann nicht äquivalent zu einem nicht Akzeptierzustand sein

1. Der Reihe nach alle leeren Felder kontrollieren:

Wenn man von einem Paar (qi, qj) mit einem bestimmten Zeichen-Übergang zu einem Paar kommt, welches bereits mit (x) markiert ist, ist auch (qi, qj) nicht äquivalent (x)

1. 5. Schritt wiederholen, bis keine neuen (x) mehr gesetzt werden können
2. Alle restlichen Paare können zusammengesetzt werden und markieren (≡)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | q0 | q1 | q2 | q3 |
| q0 | ≡ | **x** | **x** | **x** |
| q1 | **x** | ≡ | ≡ | **x** |
| q2 | **x** | ≡ | ≡ | **x** |
| q3 | **x** | **x** | **x** | ≡ |

**A picture containing clock, watch

Description automatically generatedBeispiel:**

## pumping lemma

Mit dem Pumping Lemma kann man bestimmen, ob eine Sprache regulär ist, ohne alle Zustände aufzulisten. Dazu wird ein Widerspruchsbeweis durchgeführt. Eigenschaften des Pumping Lemmas für ein Wort aufgeteilt in sind:

**Beispiel:**  (Gleich viele 0 wie 1)

1. Annahme: Sprache L regulär ist
2. , Pumping Length
3. Wort wählen mit

1. Unterteilung

Diagram

Description automatically generated

1. Pumpen: Nur die Anzahl von 0 wird erhöht, Anzahl 1 bleibt

A picture containing graphical user interface

Description automatically generated

1. für , im Widerspruch zum Pumping Lemma

# Nea

## definition

Übergange zwischen Zuständen in einem NEA sind nicht eindeutig. Ein NEA besitzt das gleiche Quintupel wie ein DEA.

**Beispiel:**

A picture containing text, clock

Description automatically generated

A picture containing qr code

Description automatically generated

## NEA → DEA transformation

1. Alle möglichen Zustände inkl. Komplementärzustände aufzeichnen
2. Für alle Elemente des Alphabets einen Übergang einzeichen

→ Kein Übergang ist auch ein Übergang

1. A picture containing text, watch

   Description automatically generatedDiagram

   Description automatically generatedNicht erreichbare Zustände können weggelassen werden

# reguläre ausdrücke

## ausdrücke

|  |  |  |
| --- | --- | --- |
| Ausdruck r | L(r) | Bedeutung |
| . | ∑ | Beliebiges Zeichen ∈ ∑ |
| a | {a} | Zeichen a ∈ ∑ |
| a\* | {a\*} | 0 oder beliebig viele a |
| ab+ | {aa\*} | a und mindestens 1 b |
| a? oder |a |  | 0 oder 1 mal a |
| a{5} | {a5} | Genau 5 mal a |
| a{2,} | {aaa\*} | 2 oder mehr a |
| a{,4} |  | Maximal 4 a |
| a{1,3} |  | Zwischen 1 bis 3 a |
| ab | cd |  | ab oder cd |
| [abc] | {a, b, c} | Ein Zeichen aus {...} ∈ ∑ |
| [1-9] | {1, 2, ..., 9} | Positive Ziffer |
| [^abc] |  | Nicht a, b oder c |
|  | {ε} | Leeres Wort |
|  | {} | Leere Sprache |

**Beispiele:**

Daten der Form dd/mm/yyyy = ([0-9]{1,2}) / ([0-9]{1,2}) / (0-9){4}

Zurich (ZRH) Switzerland = [A-Z][A-Za-z ]\* ([A-Z]{3}) [A-Z][A-Za-z ]\*

## operationen

**Alternative:**

Neuen Startzustand erstellen und mittels -Übergang mit den ursprünglichen Startzuständen verbinden.

A picture containing chart

Description automatically generated

**Verkettung:**

Akzeptierzustände mit Startzusatnd mit -Übergang verbinden.

**Diagram, schematic

Description automatically generated**

**\*-Operation:**

S

Akzeptierzustände von A1 mit einem neu erstellten Akzeptierzustand verbinden. Dieser neue Akzeptierzustand mit ε -Übergang mit dem Startzustand des Automaten verbinden.

Shape, circle

Description automatically generated with medium confidence

## Regex → nea

1. Regex von innen nach aussen übersetzen (Klammern zuerst)
2. Einzelne Teile mittels ε-Übergänge verbinden

|  |  |
| --- | --- |
| a |  |
| ∑ |  |
| ε |  |
| ∅ |  |
| ab|cd |  |
| (ab)\*c |  |
| (ab)+ |  |

## vnea

Ein VNEA ist ein NEA welcher aus zwei Zuständen besteht mit einem einzigen Übergang welcher mit regulären Ausdrücken beschrieben ist.

Diagram, shape

Description automatically generated**Beispiel:** DEA in einen äquivalenten Ausdruck umwandeln

1. Neuer Start- und Akzeptierzustand hinzufügen

A picture containing watch, scissors

Description automatically generated

1. Zustand 1 entfernen und Pfade S-1-2, S-1-3, 2-1-3, 3-1-2 und 2-1-2 anpassen

Diagram

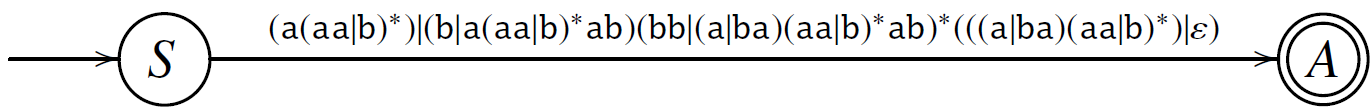
Description automatically generated

1. Zustand 2 entfernen und Pfade S-2-A, S-2-3, 3-2-A und 3-2-3 anpassen

Diagram

Description automatically generated

1. Zustand 3 entfernen



# kontextfreie sprache

Eine kontextfreie Grammatik ist ein Quadrupel mit

1. V : Endliche Menge von Variablen
2. : Endliche Menge von Zeichen disjunkt zu V (Terminalsymbole)
3. R : Menge von Regeln bestehend aus Variable und Kette von V oder
4. S : Startzustand

**Beispiel:** V = {A}, = {(, )}, S = A, R = { }

Buchtabe A steht nicht immer für das gleiche, er ist Platzhalter für einen korrekten Klammerausdruck. A ist daher eine Variable. Die Zeichen ( und ) können nicht weiter ersetzt werden und werden daher Terminalsymbole genannt.

Kontextfrei kommt davon, dass bei den Regeln nicht auf den Kontext ankommt, in dem eine Variable auf der linken Seite des Pfeils vorkommt. Regeln der Form sind kontextsensitive Regeln. Solche Regeln sollen jedoch nicht zugelassen sein.

Text, letter

Description automatically generated**Menge der Wörter aus Grammatik G:**

## parse tree

Ein Parse Tree ist der Ableitungsbaum eines Wortes einer kontextfreien Sprache in Baum-Struktur.

Zwei Ableitungen eines Wortes w einer kontextfreien Sprache L(G) heissen äquivalent, wenn sie den gleichen Ableitungsbaum haben. Hat eine Sprache Wörter mit verschiendenen Ableitungen, heisst sie mehrdeutig.

**Beispiel:**

A picture containing boat, water

Description automatically generatedA picture containing boat, water, outdoor, shore

Description automatically generated

## chomsky normalform (cnf)

Eine kontextfreie Grammatik ist ein Chomsky Normalform wenn auf der rechten Seite nicht

vorkommt und jede Regel von der Form oder ist, zusätzlich ist die

Regel erlaubt.

1. Damit Startvariable nicht auf rechten Seite auftritt, wird eine neue Startvariable hinzugefügt, sowie eine neue Regel , wobei die ursprüngliche Starvariable war.
2. A close-up of a calculator

   Description automatically generated with low confidenceEntfernung aller -Regeln. Für jede Regel, die auf der rechten Seite eine solche Variable entfällt, fügen wir eine zusätzliche Regeln hinzu, bei der die -Regeln angewendet worden ist.
3. Entfernung von sogennanten «unit rules».Regeln mit einer einzelnen Variablen auf der rechten Seite können dadurch eliminiert werden, dass man sie in jeder Regel anwendet, die auf der linken Seite entfällt.

Ursprünliche Regel behalten, denn B könnte ja auch auf eine andere Art erhalten worden sein als mit der Regel .

1. Text, letter

   Description automatically generatedVerkettung: Regel der Form kann mit Hilfe neuer Variablen in folgende Regeln abgebildet werden.   
   Falls  ein Terminalsybol ist, ersetzen wir in der obigen Regeln durch eine neue Variable und fügen die Regel hinzu.

**Beispiel:**

Diagram

Description automatically generated with medium confidence

## cyk-algorithmus (cocke younger kasami)

Mit dem CYK-Algorithmus kann ermittelt werden ob ein Wort zur kontextfreien Sprache gehört.

1. Grammatik G muss in Chomsky-Normalform sein
2. Diagram

   Description automatically generated with medium confidenceRekursiv prüfen ob sich eine Variable in ein Terminalsymbol oder das leere Wort überführen lässt.

## stackautomat

Ein Stackautomat ist ein Sextuplet mit

1. Q : Zustände
2. : Eingabe-Alphabet
3. : Stack-Alphabet
4. : heisst Übergangsfunktion
5. : Startzustand
6. : Menge der Akzeptierzustände

Stackautomat nicht immer deterministisch möglich

**Bedingung:** Ist L eine kontextfreie Sprache mit Grammatik G, gibt es einen Stackautomat der L akzeptiert. L muss in der Chomsky Normalform sein.

**A picture containing text, gauge, device

Description automatically generatedÜbergänge:** Verarbeite Input a und ersetzte b auf dem Stack durch c

**Stack:** Ein unendlich grosser Speicher, bei der immer nur das oberste Element sichtbar ist.

A picture containing line chart

Description automatically generatedDiagram

Description automatically generated**Beispiel:**

**Beispiel:**

Diagram

Description automatically generated

**Grundlegende Operationen:**

**Diagram

Description automatically generated**

**-Operationen:**

1. a,c = c wird auf den Stack gelegt
2. a, b= b wird vom Stack entfernt
3. a, = Stack bleibt unverändert

## pumping lemma für cfg

Ist L eine kontextfreie Sprache, dann gibt es eine Zahl N, die Pumping Length, derart, dass jedes Wort mit zerlegt werden kann in 5 Teile w = uvxyz sodass



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

**Beispiel:**

1. Annahme: L ist kontextfrei
2. Pumping length N
3. Wort
4. A screenshot of a computer

   Description automatically generated with low confidenceZerlegung:
5. Beim Pumpen nimmt die Anzahl der a und b zu, aber nicht die Anzahl der c
6. Widerspruch: L nicht kontextfrei

## stackautomat → Grammatik

1. Icon, circle

   Description automatically generatedNur ein Akzeptierzustand: neuer Akzeptierzustand und Übergänge
2. Chart, scatter chart

   Description automatically generatedStack leeren:
3. Diagram

   Description automatically generatedJeder Übergang legt entweder ein Zeichen auf den Stack oder entferntn eines:
4. Grammatik erstellen
   1. Variablen der Grammatik sind Übergänge von Zustand p in Zustand q
   2. Diagram, schematic

      Description automatically generatedA picture containing text

      Description automatically generatedDiagram

      Description automatically generatedChart, line chart

      Description automatically generatedBeginnend mit Start- und Endzustand schauen, was für Zeichen auf den Stack gelegt und vom Stack entfernt werden. Dabei werden 2 Fälle unterschieden.  
      **Fall 1:** Zeichen unterschieden sich. Zeichen muss früher bereits vom Stack entfernt worden sein **Fall 2:** Zeichen sind gleich.

Diagram

Description automatically generated**Beispiel:**

Startvariable:

Übergänge die das Zeichen $ behandeln führen zu

Wenn man in eine 0 auf den Stack legt, muss man

in wieder entfernen

Man kann aber auch ohne eine 0 von zu

gelangen mit

Zusätzlich haben wir , aber da diese nur in Regel 3 verwendet wird können wir diese ersetzen mit .

Die Grammatik sieht dann so aus:

Es werden noch 2 Variablen verwendet, aber die Startvariable wird immer in umgewandelt, man kann sie also einsparen. Schreiben wir bleibt somit

# turing machine

Diagram

Description automatically generatedEine Turing Maschine ist ein Septuplet mit

1. Icon

   Description automatically generatedQ : Zustände
2. : Alphabet
3. : Bandalphabet
4. :
5. : Startzustand
6. : Akzeptierzustand
7. : Ablehnungszustand

Übergang möglich wenn a under dem Schreib-/Lesekopf, aktuelles Feld auf dem Band wird mit b überschrieben und der Kopf bewegt sich nach links (L) oder rechts (R)

**Arbeitsweise:**

1. Input-Wort w auf dem Band
2. Schreib-/Lesekopf positioniert auf 1. ZeichenDiagram

   Description automatically generated
3. Maschine starten, t(w) Einzelschritte ausführen
4. Maschine hält in oder an. Diagram

   Description automatically generated

**Laufzeit:**

## berechnungsgeschichte

**Notation für Maschinenzustand:**

Diagram

Description automatically generated

Zustandszeichen befindet sich immer vor der Zelle auf der der Schreiber steht.

**Übergang:**

Calendar

Description automatically generated with low confidence

Übergang möglich wenn a auf Band, wird durch b ersetzt und Kopf wird nach links gesetzt.

**Beispiel:**

A picture containing qr code

Description automatically generatedDiagram

Description automatically generated

## Varianten

**Anderes Bandalphabet:** Bandalphabet binär codieren, d.h mit Zeichen aus , simulierbar in Zeit

**Mehrere Bänder:**

A picture containing diagram

Description automatically generated

Simulierbar in Zeit

**Mehrspurige Turing-Maschine:**

**Shape

Description automatically generated**

Simulierbar in Zeit

**Nicht deterministische Turing Maschine:**

Mehrere Berechnungsmöglichkeiten vorhanden. Jede nichtdeterministische Turingmaschine ist äquivalent zu einer deterministischen Turingmaschine. Nicht deterministische TM mit Laufzeit hat in einer Standard TM eine Laufzeit von .

wenn es einen Berechnungsweg gibt, der zu führt auch wenn es Wege zu gibt.

//todo

**Aufzähler:** Ein Aufzähler ist eine TM mit einem Drucker, die auf einem leeren Band beginnt. Die vom Aufzähler ausgegebenen Wörter bilden eine Sprache, die vom Aufzähler aufgezählte Sprache.

Diagram

Description automatically generated

# Entscheidbarkeit

**Definition:** Eine Sprache L heisst entscheidbar, wenn es einen Entscheid M gibt mit

, der dazu verwendet werden kann, herauszufinden, ob ein Wort w in der Sprache liegt. Man sagt, M entscheidet L.

**Entscheider:** Ein Entscheider in eine TM die auf jedem beliebigen Input anhält

**Turing erkennbar/akeptierbar:** Ein TM akzeptiert eine Sprache L wenn die Eingabe jedes Wortes nach endlich vielen Schritten in einem akzeptierenden Zustand hält und bei in einem nicht akzeptierten Zustand oder überhaupt nicht hält.

**Turing entscheidbar:** TM akzeptiert eine Sprache und hält garantiert in endlicher Zeit, auch bei .

**Entscheidbar vs Erkennbar:** Entscheidbare Sprache hält für jeden beliebigen Input an. Erkennbare Sprache hält nur für akzeptierten Input an.

|  |  |
| --- | --- |
| Turing Erkennbar | Turing entscheidbar |
| A language which is Turing Recognizable if there is a Machine that will halt and accept only the strings in that language and not in that language, then that TM either rejects, or does not halt at all. | A language is said to be Decidable if there is a Machine that will accept strings in the language and reject strings not in the language. |
| A Language is called Turing Recognizable if some Turing Machine recognizes it. | A Language is called Turing Decidable if some Turing Machine decides it. |
| If there exists a Turing Machine such that when encountering a string in that language, the machine terminates and accepts that string then we can say that type of language is a Turing recognizable. | If there exists a Turing Machine such that when encountering a string in that language, the machine terminates and accepts that string then we say that type of language is Turing decidable. |
| If there exists a Turing Machine such that when encountering a string not in that language, the machine either terminates and rejects that string or doesn’t terminate at all then we can say it is Turing-Recognizable. | If there exists a Turing Machine such that when encountering a string not in that language, the machine terminates and rejects that string then we can say it is Turing-Decidable. |
| It is not stronger condition than Turing Decidable | It is a stronger condition than Turing-Recognizable. |

## reduktion

**Definition:** Mittels Reduktion kann bewiesen werden, dass eine TM nicht eintscheidbar ist. Man reduziert dabei eine TM auf ein nicht entscheidbares TM.

A picture containing qr code

Description automatically generatedDiagram

Description automatically generated

**Notation:**

: A leichter entscheidbar als B

B entscheidbar, A entscheidbar

A nicht entscheidbar, B nicht entscheidbar

Diagram, venn diagram

Description automatically generated

## entscheidbarkeits-probleme

E = Leerheitsproblem, EQ = Gleichheitsproblem, A = Akzeptanzproblem

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Problem | Wort | Bedingung | Ents. | Entscheidbarkeitsalgorithmus |
|  |  |  | ja | Minimalautomat hat keinen Akzeptierzustand |
|  |  |  | ja | Chomsky-Normalform |
|  |  |  | nein |  |
|  |  |  | ja | Vergleich der minimalen Automaten |
|  |  |  | nein |  |
|  |  |  | nein |  |
|  |  |  | ja | Regex-Engine simulieren beliebige DEA auf beliebigen Input w |
|  |  |  | ja | Cocke-Younger-Kasami deterministischer Parse-Algorithmus |
|  |  |  | nein | Halteproblem |

## leerheitsproblem

**DEA**:

entscheidbar

Herausfinden ob überhaupt je ein Akzeptierzustand erreicht werden kann.

1. Starzustand markieren
2. Solange sich noch neue Zustände markieren lassen: Markiere alle Zustände, die sich von den bereits markierten aus erreichen lassen
3. Falls Akzeptierzustand markiert: , andernfalls

**CFG:**

entscheidbar

Eine Grammatik erzeugt Wörter, wenn die Startvariable am Ende zu einem Terminalsymbol abgeleitet werden kann.

1. Wandle G in Chomsky Normalform um
2. Markiere alle Terminalsymbole
3. Solange sich neue Variablen markieren lassen: Markiere alle Variablen A, zu denen es eine Regel gibt, so dass alle Symbole auf der rechten Seite der Regel bereits markiert sind.
4. Falls S markiert wurde: , andernfalls

**TM:** ,

nicht entscheidbar + Reduktion

Akzeptiert die Maschine M das Wort w? Es gibt keinen EntscheiderGraphical user interface, text, application

Description automatically generated

## gleichheitsproblem

**DEA**:

entscheidbar

1. Erzeuge minimalen Automaten A’ zu A
2. Erzeuge minimalen Automaten B’ zu B
3. Falls A’ = B’ dann sonst

**CFG**:

nicht entscheidbar + Reduktion

**TM:**

nicht entscheidbar + Reduktion

## akzeptierproblem

**DEA:**

entscheidbar

1. Simuliere A auf einer TM mit Input w
2. Falls A akzeptiert, sonst

**CFG:**

entscheidbar

1. Wandle G in Chomsky Normalform um
2. Erzeuge alle Wörter, die sich durch Anwendung von - 1 Regeln der Form und Regeln der Form ableiten lassen
3. Fallsl w darunter vorkommt, sonst oder CYK-Algorithmus

**TM:**

nicht entscheidbar + Reduktion

Auf den ersten Blick ist dieses Resultat sehr überraschen, warum soll für TM alles anders sein, als für endliche Automaten/Stackautomaten? Der wesentliche Unterschied verbirgt sich im Wort «entscheidbar» selbst. Ein Entscheider ist eine TM, welche in diesem Fall Aussagen über andere TMs machen muss, also zum Beispiel auch über sich selbst.

Graphical user interface, text, application

Description automatically generated

**Bsp:**

Diagram, schematic

Description automatically generated

## halteproblem

**TM:**

nicht entscheidbar

Reduktion c konstruieren. Die Reduktion muss aus einer TM M und einem Input w eine neue TM S und ein Input w machen, sodass S auf w genau dann hält, wenn M das Wort w akzeptiert.

Diagram

Description automatically generated

## satz von rice

Der Satz von Rice besagt ob eine Sprache nicht entscheidbar ist. P sei eine Eigenschaft der Turing erkennbaren Sprache L, wobei es eine Sprache L1 gibt, welche die Eigenschaft P besitzt und eine Sprache L2, welche die Eigenschaft nicht hat.

**Trivial:** Eine Eigenschaft P Turing-erkennbarer Sprache heisst f, wenn es zwei Turing-Maschinen gibt von welcher nur eine davon die Eigenschaft P besitzt.

**Nichttrivial:** P heisst nicht trivial, falls es eine Sprache L1 gibt, die die Eigenschaft P hat, und eine Sprache L2, die die Eigenschaft nicht hat.

1. Nichttriviale Eigenschaft P aufschreiben
2. Beide Sprachen L1 und L2 bilden
3. Gibt es ein Programm, welches beide Sprachen erkennen kann? Sind beide Sprachen Turing erkennbar?
4. Falls nein: Dann besagt der Satz von Rice, dass die Sprache nicht entscheidbar ist.

**Beispiel:** Gibt es ein Programm, welches Primzahlprüfer auf Korrektheit testen kann?

1. P: Die akzeptierte Sprache ist die Menge der Primzahlen
2. L1: Primitiver Primzahlenprüfer der alle möglichen Teiler durchprobiert akzeptiert genau die Primzahlen, die akzeptierte Sprache halt also die Eigenschaft P  
   L2: Leere Sprache ist Turing-erkennbar, hat aber die Eigenschaft P nicht
3. Es gibt kein Programm, welches die beiden Sprachen erkennen kann   
    P ist nichttrivial
4. Nach Satz von Rice ist P nicht entscheidbar

# komplexität

Entscheidbare Probleme mit exorbitant langer Laufzeit sind praktisch auch nicht lösbar. Zb: ).

Diagram

Description automatically generated

**Klasse P:** Zu der Klasse P gehören Sprachen die mit einem Entscheider in polynomieller Laufzeit (nk) entschieden werden können (Entscheider = TM die garantiert anhält, also deterministische TM) Bsp: Kontextfreie Sprachen, Teilerfremdheit

**Klasse NP:** NP ist genau dann die Klasse der Sprachen, für die ein deterministischer polynomieller Verifizierer existiert. Zu der Klasse NP gehören Sprachen, die in polynomieller Zeit von einer nichtdeterministischen Turing Maschine entschieden werden können.

**NP-Vollständig:** Schwerste Probleme in NP (siehe Katalog von Karp)

**Verifizierer:** Ein polynomieller Verifiziere für die Sprache L ist eine TM V, so dass es für jedes ein Wort c (das Lösungszertifikat) gibt, für das gilt .

Laufzeit von V ist polynomiell in .

A picture containing text, crossword puzzle

Description automatically generatedGraphical user interface, text, application, email

Description automatically generated

## polynomielle reduktion

Polynomieller Laufzeit-Vergleich: Seien A und B entscheidbar Sprachen.

Eine berechenbare Abbildung mit

1. (Reduktion)
2. ist berechenbar in polynomieller Zeit in

heisst polynomielle Reduktion . Lies: A ist polynomiess leichter entscheidbar als B.

Sei . Dann gilt:

A picture containing text, clock, gauge, watch

Description automatically generated

Chart, radar chart

Description automatically generated

# katalog von karp

**Vorgehen:**

1. Gewähltes Problem aus dem Katalog von Karp notieren
2. Reduktion des gegebenen Problems, auf das bekannte aus Karp
3. Beschreibung der Reduktion
4. Schlussfolgerung: Da das Problem aus dem Katalog von Karp NP-vollständig ist, gibt es auch keinen effizienten (polynomiellen) Algorithmus, der das gegebene Problem lösen könnte.

**Schlusssatz:** Dies ist genau die Beschreibung des Problems HMMM. Das Problem HMMM ist NP-vollständig, nach aktuellem Wissen gibt es also keinen effizienten (polynomiellen) Algorithmus, der ein HMMM Problem lösen könnte.

Chart, radar chart

Description automatically generated

## sat (sATISFIABILITY)

SAT =

Für eine logische Formel testet man einfach alle möglichen Belegungen der Variablen mit Wahrheits-werten. Dafür sind 2n Verifikationen notwendig. Der Verifizierer verlangt als Lösungszertifikat c eine Belegung der Variablen mit Wahrheitswerten und überprüft. Ob durch einsetzten der Werte die Formel wahr wird.

**Beispiel Crazy-Elektriker:**

In einem Labyrinth gibt es Räume mit mind. einer Lampe. Am Eingang befindet sich ein Schaltet. Jeder Schaltet ist ein Umschalter, er ist mit zwei Stromkreisen verbunden. Jeder Stromkreis ist mit Lampen in den einzelnen Räumen verbunden (nicht unbedingt mit allen). Umlegen des Schalters: Lampen erstes Stromkreis an, Lampen zweiter Stromkreis aus

Stromkreis ↔ Beider Werte von xi (x und x̄)

Lampen ↔ EIN wenn x, AUS wen x̄

Schalter ↔ xi

Raum ↔ Logische Formel

Für alle Räume:

**Schlussfolgerung:** Da SAT NP-vollständig ist, ist auch das Crazy-Elektriker-Problem NP-vollständig

## 3sat (Enthält Zahl 3)

3SAT ist eine Variante von SAT. Logische Formel in Konjunktiver Normalform und maximal 3 Variablen in jeder Klausel.

## k-clique (enthält k Z)

Eine k-Clique in einem Graph G ist eine Menge von k Ecken des Graphen so, dass in G jede Ecke der Teilmenge mit jeder anderen Ecke verbunden ist. Im umgangssprachlichen Gebrauch ist eine Clique eine Gruppe von Leuten, in der jeder jeden kennt.

A picture containing diagram

Description automatically generated

**Beispiel Job-Parallelisierbarkeit:**

Gegeben sei eine Menge von n Jobs  die jeweils exklusiv auf m Ressourcen zugreifen, diese 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.

Jobs ↔ Ecken (Knoten)

Jobs die gleichzeitig laufen können ↔ Kanten

Prozessoren ↔ k, Grösse der CLIQUE

## bip (binary integer programming)

BIP ist “binary integer programming”, zu einer ganzzahligen Matrix C und einem ganzzahligen Vektor d ist ein binärer Vektor x zu finden mit

C x d

## Set-packing (enthält k Z)

Gegen sei eine Familie  von Mengen undn eine Zahl . Gibt es eine k-elementige Teilfamilie mit , derart, dass die Mengen der Teilfamilie paarweise disjunkt sind, also

Einfacher: Gegeben eine Familie von Mengen und eine Zahl k. Gesucht sind k Mengen in dieser Familie, welche paarweise disjunkt(keine gemeinsamen Elemente) sind und ebenfalls eine Familie bilden.

**Beispiel Arzneitest:**

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 (Pollen, Katzenhaare, Hausstaub, Lactose, …) er allergisch reagiert. Die Untersuchung soll sich auf eine Teilmenge von k = 17 oder noch mehr ausgewählten Allergenen beschränken, die so beschränken ist, dass kein Proband auf mehr als eines der ausgewählten Allergene reagiert. Es stellt sich als schwierig heraus, eine solche Teilmenge zu finden. Warum?

Allergene ↔

Auf Allergen i allergische Probanden ↔

Ausgewählte Allergene ↔

Ausschlussbedingung zwischen Allergenen i und j ↔

Es wird verlangt, k Allergene auszuwählen, also eine Teilmenge zu finden.

## Set-covering (enthält k mengen)

Gegeben eine endliche Familie endlicher Mengen und eine Zahl k, gibt es eine Unterfamilie bestehend aus k Mengen, die die gleiche Vereinigung hat?

Text

Description automatically generated

**Beispiel:**

Um diesem Unheil ein Ende zu setzen, kamen die Stämme überein, den Ältestenrat zu bilden, in dem wenn auch nicht jeder Stamm direkt, so doch mindestens das “Blut” jedes Stammes vertreten sein musste. Damit trug man der Tatsache Rechnung, dass Mobiltät und planetenweiter Handel mittlerweile zu einer weitgehenden Durchmischung in der Bevölkerung geführt hatte. Ausgeprägtes Traditionsbewusstsein hatten jedoch sichergestellt, dass von jedem Bürger bekannt war, aus welchen Stämmen er “Blut” in sich trug. Warum ist trotzdem eine weit entwickelte Technologie nötig, um herauszufinden, ob sich ein Ältestenrat mit k Mitgliedern überhaupt bilden lässt?

**Lösung:**

Das Problem, einen Ältestenrat mit Mitgliedern zu bilden, ist äquivalent zum Set-Covering Problem. Zu jedem Bürger ist die Menge aller Stämme bekannt, von denen er “Blut” in sich trägt. Die Vereinigung aller ist die Menge aller Stämme. Gefragt wird jetzt nach Bürgern so, dass die bereits alle Stämme abdecken, also .

Die Übersetzung von Set-Covering auf das Ältestenratsproblem ist also

↔ Menge aller Stämme

↔ Bürger

↔ Menge der Stämme, die durch i vertreten werden können

Unterfamilie ↔ Ältestenrat

## exact-cover (disj Teilmen, gleich verein.)

Gegeben eine Familie von Teilmengen einer Menge U, gibt es eine Unterfamilie von Mengen, die disjunkt sind, aber die gleiche Vereinigung haben? Die Unterfamilie muss also muss also und erfüllen.

**Beispiel:**

Gegeben sind eine Menge X und eine Menge S welche aus nichtleeren Teilmengen von X besteht. Gesucht ist eine Teilmenge U von S, deren disjunkte Vereinigung X ist. => Das heisst jedes Element in X soll genau in einer der Mengen von U vorkommen:

X = {a,b,c,d,e,f},

S={{a,b},{a,b,c},{c,e},{d,f},{e,f}},

U={{a,b},{c,e},{d,f}}

Gibt es eine minimale Anzahl Gruppen so dass die gesamte Menge abgedeckt ist?

**Beispiel Big Data Shop-Webseite:**

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 derart gibt, dass jeder Kunde genau eine der Eigenschaften hat. Die Abgabe der Arbeit steht in zwei Tagen bevor, und er hat noch keinen funktionierenden Algorithmus. Muss er sich Sorgen machen?

Eigenschaft ↔ Menge

Teilmenge von Eigenschaften ↔ Unterfamilie

Genau eine der Eigenschaften ↔

## clique-cover (enthält positive 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? (Wichtig: Die Ecken müssen sich kennen / miteinander verbunden sein, durch den AMG, da Clique) Kann der Graph in mindestens k Gruppen aufgeteilt werden. Wobei sich alle Gruppenteilnehmer kennen müssen.

**Beispiel Gruppenarbeit:**

Radar chart

Description automatically generated with medium confidenceFür eine Gruppenarbeit sollen k Gruppen gebildet werden. Um die Zeit für das gegenseitige Kennenlernen möglichst kurz zu halten, sollen sich die Leute einer Gruppe bereits gegenseitig kennen. Alle Leute sollen beschäftigt sein. Können Sie einen effizienten Algorithmus formulieren, mit dem eine solche Gruppeneinteilung auch bei einer grossen Teilnehmerzahl gefunden werden kann?

Teilnehmer ↔ Knoten

Kennen sich ↔ Kante

Anzahl Gruppen ↔ k

Gruppe ↔ Clique

## vertex-cover (enthält k vertices)

A picture containing clock

Description automatically generatedGegeben ein Graph G und eine Zahl k, gibt es eine Teilmenge von k Vertices so, dass jede Kante des Graphen ein Ende in dieser Teilmenge hat?

Beispiel: k=3

Führt jede Kante des Graphen g auf einen Knoten aus U,

welche eine Teilmenge aus maximal k Knoten ist.

**Beispiel Telefon Anschlüsse abhören:**

Die Software kann mit maximal k Anschlüssen umgehen, die abgehört werden können. Diktator Xetrev befiehlt daher, dass innert einer Woche eine Menge von k Anschlüssen definiert werden müssen, so dass jedes Telefongespräch abgehört werden kann.

Das Problem, welches die Experten lösen sollten, ist das Problem VERTEX-COVER. Die Telefonanschlüsse bilden die Knoten eines Graphen, die Kanten verbinden diejenigen Anschlüsse, die tatsächlich miteinander telefonieren. Gesucht ist eine Menge von k Knoten so, dass jedes mögliche Gespräch abgehört werden kann, also jede Kante in einem ausgewählten Knoten endet.

**Beispiel:**

Ein Verkehrsnetz soll regelmässig durch Mitarbeiter kontrolliert werden, die ihre Basis an einzelnen Knotenpunkten des Netzes haben. Kann man auf effiziente Art und Weise herausfinden, an welchen Knotenpunkten man Kontrolleure stationieren muss, damit jede Strecke in einem Knoten mit Kontrolleur endet?

**Lösung:**

Dieses Problem is äquivalent zu VERTEX-COVER, und damit NP-vollständig, also nach aktuellem Wissen für grosse Probleme nicht effizient lösbar. Die Reduktionsabbildung

bildet die Objekte wie folgt ab

Knoten ↔ Knoten

Strecke ↔ Kante

Anzahl Kontrolleure ↔ k

Knoten mit Kontrolleur ↔ Knoten aus dem Vertex-Cover

## Hitting-set (Untermen in k anz vertretet)

Text, letter

Description automatically generatedGegeben eine Menge von Teilmengen , gibt es eine Menge H, die jede Menge in genau einem Punkt trifft, also ?

**Einfacher:**

Gegeben ist eine Menge von Teilmengen S eines „Universums“

A picture containing clipart

Description automatically generatedT, gesucht ist eine Teilmenge H von T so, dass jede Menge in S

mindestens ein Element aus H enthält. Zusätzlich ist gefordert,

dass die Anzahl der Elemente von H einen gegebenen Wert k

nicht überschreitet. Anders gesagt gegeben ist eine Menge von

Setz Suche ein neues Set welches 1 Element aus jedem

beinhaltet ohne Wiederholung

**Beispiel Expertenkommission:**

Aus einer Menge von Fachleuten, die zum Teil in mehreren Gebieten tätig sind, soll eine Expertenkommission gebildet werden. Da zwei Experten für das gleiche Fachgebiet sich erfahrungsgemäss immer streiten, will man in der Expertenkommission jedes Fachgebiet durch genau einen Experten vertreten haben. Können Sie einen effizienten Algorithmus zur Auswahl der Mitglieder der Kommission angeben?

Experte ↔ Punkt in S

Fachgebiet i ↔ Teilmenge Ui aller Experten für dieses Gebiet

Expertenkommission ↔ Hitting Set H

A picture containing qr code

Description automatically generated

## Partition

Gegeben ein Folge von s ganzen Zahlen , kann man die Indizes in zwei Teilmengen I und Ī teilen, so dass die Summe der zugehörigen identisch ist:

**Einfacher:** Gegeben eine Menge von ganzen Zahlen. Gesucht sind 2 disjunkte Mengen, welche als Vereinigungsmenge wieder die Ausgangsmenge ergeben und die Summe aller Zahlen in der einen Menge gleich der Summe der anderen Menge ist.

Text

Description automatically generated

**Beispiel toter König:**

Ein reicher König ist gestorben, und hinterlässt seinen beiden Söhnen, eineiigen Zwillingen, ein grosses Vermögen aus Schlössern und Burgen. Ist eine gerechte Teilung möglich?

## hampath (gerichteter graph)

Geschlossener Pfad in einem gerichteten Graphen, der JEDEN Vertex des Graphen GENAU EINMAL besucht. (muss sich nicht im Kreis bewegen, sonst ist es ein HAMCYCLE)

**Beispiel CEO Personalbesuch:**

Der neue CEO einer grossen Fluggesellschaft möchte das Personal in allen Flughäfen persönlich besuchen, die die Fluggesellschaft anfliegt. Er bittet seine Sekretärin, einen optimalen Besuchsplan zusammenzustellen, bei der er jede Destination nur genau einmal

besuchen muss.

**Lösung:**

Dieses Problem entspricht dem HAMPATH Problem. Dieses Problem ist NP-vollständig, es gibt also keine effizienten Algorithmen für grosse solche Probleme.

A picture containing mirror

Description automatically generatedHAMPATH ≤p BESUCH

Knoten ↔ Destination

Kante ↔ Flug

Hamiltonischer Pfad ↔ Besuchsplan

## uhampath (ungerichteter graph)

UHAMPATH ist das Problem in einem ungerichteten Graphen einen hamiltonschen Pfad zu finden.

**Beispiel Facebook Freunde:**

Facebook hat einige hundert Millionen aktiver Mitglieder. Jeder

Teilnehmer kann mit jedem anderen Teilnehmer befreundet sein oder auch nicht. Wie aufwendig ist es, die Liste aller Freunde eines Teilnehmers so zu sortieren, dass zwei aufeinanderfolgende Freunde in der Liste untereinander ebenfalls befreundet sind?

A picture containing mirror, scissors

Description automatically generatedUHAMPATH ≤p FACEBOOK

Vertizes ↔ Teilnehmer

Kanten ↔ Befreundete Teilnehmer

## feedback-node-set (enthält k vertices)

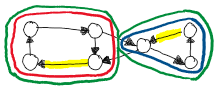
Gegeben gerichteter Graph G und eine Zahl k, gibt es eine endliche Teilmenge von k Vertices von G so, dass jeder Zyklus in G einen Vertex in der Teilmenge enthält?

**Beispiel Roboter prüfen Punkte:**

Das Budget reicht für die Ausrüstung von k Prüfstellen. Die Roboter bewegen sich auf dem

gerichteten Graphen der Bodenmarkierungen. Sie fahren die Zyklen des Graphen ab. Gesucht wird jetzt eine Menge von k Netzknoten für die Prüfstellen, so dass jeder Zyklus des Graphen durch eine dieser Prüfstellen verläuft.

## feedback-arc-set (enthält k kanten)

Gegeben ein gerichteter Graph G und eine Zahl k, gibt es eine Teilmenge von k Kanten so, dass jeder Zyklus in G eine Kante aus der Teilmenge enthält?

**Beispiel Disinfektionsstationen:**

Die Marketing-Abteilung legt fest, in welcher Richtung die einzelnen Gänge zwischen den Gestellen durchlaufen werden kann. Es soll nämlich sichergestellt werden, dass die Kunden immer noch in der “richtigen” Reihenfolge mit überflüssiger Werbung zu genauso überflüssigen Spontankäufen verleitet werden. Der Sicherheitsverantwortliche wissen, ob es möglich ist, in einigen Gängen Desinfektionsstationen aufzustellen so, dass jeder Kunde, der sich auf einem geschlossenen Weg durch das Einkaufszentrum bewegt, mindestens einmal an einer Desinfektionsstation vorbeikommt. Dabei dürfen nicht mehr Desinfektionsstationen verwendet werden, als das vorgegebene Budget erlaubt. Dies stürtzt die Planer in eine Krise, auch nach stundenlangem Probieren können Sie keine definitive Antwort geben. Warum?

Graph ↔ Plan des Einkaufszentrums

Knoten ↔ Kreuzungsstellen

Kanten ↔ Gänge

Richtung ↔ Einbahnrichtung in jedem Gang

Anzahl k ↔ Budget für Desinfektionsstationen

Arc set ↔ Platzierung der Desinfektionsstationen

Das gestellte Problem ist also NP-vollständig.

## Max-cut (Enthält aufteilung in 2 teilmen)

Gegeben ein Graph G mit einer Gewichtsfunktion und eine ganze Zahl W. Gibt es eine Teilmenge S der Vertizes, so dass das Gesamtgewicht der Kanten, die S mit seinem Komplement verbinden, mindestens so gross ist wie W:

**Einfacher:**

Graph aufteilen durchkappen von Verbindungen Resultierende Netzwerke solle möglichst hoch kumulierte Verbindungsbetrag haben.

**Beispiel Firma Aufteilung:**

Feindliche Übernahme einer Firma, mit resultierender Aufteilung der Abteilung, dass diese möglich ineffizient miteinander kommunizieren können.

Abteilung ↔ Vertex

Kommunikationsbeziehung ↔ Kante

Kommunikationsvolumen ↔ Gewicht einer Kante

Die neuen Firmeneigentümer wollen die Menge der Vertices so in zwei Mengen aufteilen, dass die Summe der Gewichte der Kanten, die durch die Aufteilung zerschnitten werden, möglichst gross wird.

## steiner-tree (Enthält gewicht < k)

Gegeben ein Graph G, eine Teilmenge R von Vertices, und eine Gewichts-funktion und eine positive Zahl k, gibt es einen Baum mit Gewicht ≤ k, dessen Knoten in R enthalten sind? Das Gewicht des Baumes ist die Summe der Gewichte über alle Kanten im Baum.

**Einfacher:** Gegeben ist ein ungerichteter Graph G, eine Menge Vertices T und eine Zahl k. Ausserdem liegt eine Funktion vor, welche jeder Kante von G einen reellen Zahlenwert zuweist. Dieser Zahlenwert ist das Gewicht der Kante. Gesucht ist nun ein Baum mit einem Gewicht von maximal k, der alle Vertices der Teilmenge T besucht.

**Beispiel:**

k = 15, T = {G, B, A}

Gewicht des Steinerbaums: 10Diagram

Description automatically generated

**Beispiel Stromnetz in Entwicklungsland:**

In einem Entwicklungsland sollen die aus dem Ausland erhaltenen Unterstützungsmittel dazu verwendet werden, endlich alle Ortschafen mit mindestens 100 Einwohnern ans Stromnetz anzuschliessen. Der Bau von Leitungen zwischen einzelnen Ortschaften ist je nach Gelände unterschiedlich teuer, zum Teil auch schlicht unmöglich. Es wird entschieden, dass man in einer ersten Phase auf Redundanz des neu zu erstellendem Netz verzichten will. Der Minister möchte endlich wissen, ob das vorhandene Geld für das Projekt ausreicht, und ist sehr ungehalten darüber, dass die Verwaltung so lange braucht, diese Frage zu beantworten. Kann man dies erklären?

Steiner Tree ≤ p Stromnetz

Knoten ↔ Ortschaften

Knoten in R ↔ zu erschliessende Ortschaften

Gewicht w einer Kante ↔ Baukosten einer Verbindungsleitung

Maximales Gewicht k ↔ Budget

## vertex-coloring (Enthält k farben)

Die Vertizes eines Graphen G können mit k Farben eingefärbt werden, so dass nie zwei durch eine Kante verbundene Vertizes die gleiche Farbe bekommen

**Beispiel Stundenplan:**

Gegeben: Fächer, Anmeldungen von Studenten für Fächer, k Zeitslots pro Woche

A picture containing sitting

Description automatically generatedGesucht: Stundenplan bzw. Zuordnung von Zeitslots zu Fächern

derart, dass jeder Student jedes angemeldete Fach besuchen kann.

Fächer ↔ Knoten

k Zeitslots pro Woche ↔ k Farben

Veranstaltung, die nicht gleichzeitig stattfindet ↔ Kante

Schlussfolgerung: Stundenplan ≤ p k-VERTEX-COLORING

## subset-sum (rucksack problem)

Gegeben ist eine Menge S von ganzen Zahlen. Kann man darin eine Teilmenge finden, die als Summe einen bestimmten Wert t hat?

SUBSET-SUM

**Beispiel: Event Budget:**

Kurz vor Jahresende in einer grossen Software-Firma: Wie jedes Jahr stellt der Gruppenleiter Software-Entwicklung fest, dass noch nicht sein ganzes Budget verbraucht ist. Er entscheidet, dass diesmal der gesamte Restbetrag bis auf den letzten Rappen ausgegeben werden soll. Daher sammelt der Gruppenleiter von seinem Team Vorschläge, was mit dem verbleibenden Geld gemacht werden könnte, und gibt die Liste seiner Sekretärin. Sie soll daraus so einige Dinge auswählen, dass genau das Restbudget verbraucht wird. Einige Tage später wundert er sich, dass die Sekretärin total überfordert ist. Ist er am richtigen Ort?

**Lösung:**

Nein, der Gruppenleiter hätte erkennen müssen, dass er der Sekretärin ein NP-vollständiges Problem gestellt hat, wofür es nach aktuellem Wissen keinen Algorithmus mit polynomieller Laufzeit gibt. Die Vorschläge des Teams bilden eine Menge von Beträgen . Daraus soll ein Teilmenge gebildet werden, so dass so dass der Restbetrag r dadurch aufgebraucht wurde:

SUBSET-SUM ≤p Event Budged

Budget ↔ Menge S (r)

Dinge / Aktivitäten ↔ Teilmenge t

## 3d-matching (enthält zahl 3)

Sei T eine endliche Menge und U eine Menge von Tripeln aus

Gibt es eine Teilmenge so, dass | und keine zwei Elemente von W in irgendeiner Koordinate übereinstimmen?

**Beispiel Frau Mann Wohnung:**

Es sind nur gewisse Frauen, Männer und Wohnungen kompatibel.

Es gibt n unverheiratete Frauen, n unverheiratete Männer, und n mögliche Wohnungen. Die ursprünglich erstellte Liste ist eine Menge von Tripeln , wobei die Zahlen x, y und z aus der Menge stammen. Aus dieser Teilmenge soll jetzt eine Teilmenge von genau n Tripeln ausgewählt werden, so dass jeder Mann, jede Frau und jede Wohnung in genau einem Tripel vorkommt. Dieses Problem ist NP-vollständig.

**Beispiel:**

Ein Koch hat je Rezepte für Vorspeisen, Hauptspeisen und Desserts. Nicht alle Vorspeisen lassen sich mit jeder Hauptspeise kombinieren, dasselbe gilt auch für Desserts. Damit jedes seiner Rezepte regelmässig zum Einsatz kommt, möchte der Koch eine Folge von Menus zusammenstellen, so dass jedes Rezept in genau einem der Menus vorkommt. Nach längerem tüfteln gibt er jedoch frustriert auf. Können Sie erklären, warum ihm die Menugestaltung so schwer gefallen ist.

**Lösung:**

Es handelt sich hier um das Problem 3D-MATCHING. Die Menge sind die Nummern der Rezepte, die Tripel aus sind die Menuzusammenstellungen, bestehend aus je einer Nummer für ein Vorspeisen-, ein Hauptspeisen- und ein Dessert-Rezept. Die Menge ist die Menge der möglichen Menukombinationen. Die gesuchte Teilmenge W ist eine Auswahl von Menus derart, dass keine Vorspeise (erste Komponente), keine Hauptspeise (zweite Komponente) und kein Dessert (dritte Komponente) mehr als einmal vorkommt. Das Problem 3D-MATCHING ist NP-vollständig, es ist daher kein Algorithmus bekannt, der das Problem in polynomieller Zeit lösen könnte.

## sequencing

Gegeben sei ein Vektor von Laufzeiten von Jobs, ein Vektor von spätesten Ausführungszeiten , einem Strafenvektor  und eine positive ganze Zahl k. Gibt es eine Permutation der Zahlen , so dass die Gesamtstrafe für verspätete Ausführung bei der Ausführung der Jobs in der Reihenfolge nicht grösser ist als ? Formal lautet die Bedingung

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Zeitverbrauch |  | fertig bis |  | Strafe |
| ½ |  |  |  | 200 |
| 3 |  |  |  | 100 |
| 50 |  |  |  | 50 |
| 20 |  |  |  | 80 |
| 1/6 |  |  |  | 100 |

Darin ist die Stufenfunktion definiert durch

**Einfacher:**

Gegeben ist eine Menge an Jobs, jeder Job hat eine Laufzeit, Ausführungszeit und eine Strafe. Die Jobs müssen sequenziell abgearbeitet werden. Wird ein Job zu spät fertig muss eine Strafe gezahlt werden. Gesucht ist eine Reihenfolge von Jobs, sodass möglichst wenig Strafe gezahlt werden muss.

**Beispiel Prüfungsvorbereitung:**

Das vorliegende Problem ist das Problem SEQUENCING, welches bekanntermassen NP vollständig ist.

Job i ↔ Vorbereitung für Fach i

Ausführungszeit ti ↔ Vorbereitungszeit ti

Deadline di ↔ Prüfungszeitpunkt di

Die Strafe für eine nicht bestandene Prüfung ist , die Reihenfolge muss so gewählt werden, dass die Gesamtstrafe .

**Beispiel:**

Um die Betriebssicherheit im Dauerbetrieb zu gewährleisten, müssen die Gleise ständig sauber gehalten werden. Dafür stehen Reinigungszüge zur Verfügung, die in regelmässigen Abständen auf vorprogrammierten Routen über die Anlage geführt werden. Natürlich sind auch noch viele andere Züge ebenfalls auf dem Schienennetz unterwegs, allerdings nicht ständig, die Loks sind Spielzeug, welches nicht für den Dauerbetrieb ausgelegt ist, sie müssen daher regelmässig Pausen zum Abkühlen einlegen. Die zeitliche Planung der Zugfahrten soll so gestaltet sein, dass innerhalb vorgegebener Zeitintervalle jeder Zug seine Fahrt durchführt. Die Zugfahrten der Reinigungszüge sind besonders wichtig. Fallen diese Fahren aus oder finden Sie zu spät statt, kann es zu Ausfällen kommen, die den Besuch des Miniatur Wunderlandes weniger attraktiv machen. Man kann sogar versuchen, den dadurch verursachten Schaden zu beziffern. Da die Reinigungszüge nicht so interessant sind wie die normalen Modellzüge, soll immer nur ein Reinigungszug aufs Mal unterwegs sein. Ziel ist natürlich, den Schaden durch Ausfälle möglichst klein zu halten.

**Lösung:**

Das Planungsproblem für die Reinigungszüge ist äquivalent zum NP-vollständigen Problem SEQUENCING, wie man mit Hilfe einer polynomiellen Reduktion zeigen kann. Dazu seien die die Fahrzeiten für jeden der Reinigungszüge, ist die späteste Fahrzeit, zu der die Zugfahrt von Zug abgeschlossen sein muss, und sind die Kosten für eine verspätete Reinigung durch den Zug .

Fahrzeit von Zug ↔Ausführungszeit von Job

Deadline für die Fahrt von Zug ↔

Kosten für Verspätete Reinigungsfahrt ↔

Reihenfolge der Fahrten ↔ Permutation π

# turing vollständigkeit

Eine (Programmier-)Sprache A ist Turing-Vollständig genau dann, wenn es eine TM-Simulator in A gibt. Eine Turing-Vollständige Sprache muss in einer Endlosschleife laufen können.

**Turing vollständige Sprachen:** C, Java, JS, Latex, WHILE, GOTO, Brainfuck etc.

TM vs moderne Computer

|  |  |
| --- | --- |
| TM | Moderne Computer |
| Zustände Q | Zuständer der CPU: Statusbits, Registerwerte |
| Band (unbegrenzter Speicher) | Virtueller Speicher: Praktisch unbegrenzter Speicher |
| Schreib-/Lesekopf | Address-Register, Programm-Zähler |
| Anhalten () | Exit(EXIT\_SUCCESS), exit(EXIT\_FAILURE) |
| Problemspezifisch | Kann beliebige Programme ausführen |

## universelle tm

Alan Turing: Es gibt eine TM, die jede beliebige andere TM simulieren kann.

**Konstruktion:**

* Eigenes Band für eine Codierung der Übergangsfunktion
* Eigenes Band für den aktuellen Zustand
* Arbeitsband
* Simulation dieser Maschine auf einer Standard-TM

**Dinge die einer TM fehlen:**

|  |  |
| --- | --- |
| Ist eine Komponente wesentlich? | Komponente verändert die Fähigkeiten einer TM oder die Komplexität auf nicht-polynomielle Weise |
| Persistenter Speicher | Files nicht unterscheidbar von Daten, die bereits im Speicher liegen |
| Interaktion | Lösung eines vollständig spezifizierten Problems braucht keine Interaktion |
| Input/Output | Output nicht wesentlich, Input existiert nicht |

**Vergleich:** TM1 ist «leistungsfähiger» als TM2, wenn TM1 TM2 simulieren kann

Mehrband TM Mehrspur TM TM mit erweitertem Alphabet TM mit {0, 1, …}

## loop

LOOP ist eine nicht Turing-vollständige Sprache da LOOP immer terminiert und daher kein Halte-Problem besitzt.

A picture containing table

Description automatically generated**Syntax:**

1. Variablen: x0, x1, x2, …
2. Konstanten: 0, 1, 2, …
3. Zuweisung: :=
4. Trennung von Anweisungen: ;
5. Operatoren: + und –
6. Schlüsselwörter: LOOP, DO, END

Programm wird so oft ausgeführt wie zu Beginn des Loopes ist

**Beispiel Summe, Produkt und IF-Statement:**

Text

Description automatically generated with low confidenceA picture containing text

Description automatically generated

IF-Statement

Text

Description automatically generatedText

Description automatically generated

## while

WHILE-Programme können zusätzlich zur LOOP-Anweisung eine WHILE-Anweisung ausführen.

**Beispiel IF:**

A picture containing text, antenna

Description automatically generated

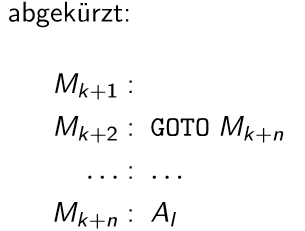
## goto

GOTO ist eine Turing-vollständige Sprache bestehend aus einer markierten Folge von Anweisungen.

**Grundelemente:**

1. Markierte Folge von Anweisungen  
   Table

   Description automatically generated with medium confidence
2. Bedingte Anweisungen  
   A picture containing text

   Description automatically generated
3. Unbedingte Anweisungen  
   A picture containing text

   Description automatically generated

**//todo mehr schreiben?**

## goto while

Die Sprachen WHILE und GOTO sind äquivalent. Ein TM Simulator kann in GOTO implementiert werden.

**GOTO in WHILE:**

A screenshot of a computer

Description automatically generated with low confidence

**WHILE in GOTO:**

Graphical user interface, text, application

Description automatically generated

# wichtige begriffe

**Alphabet**

In der Informatik ist das Alphabet meist durch die Maschine vorgegeben (sie verarbeitet zum Beispiel einzelne Bytes, also Zahlen zwischen 0 und 255). Für die Theorie von Automaten und Sprachen gilt jede nichtleere endliche Menge als Alphabet.

Bezeichnet wird ein Alphabet häufig mit grossen griechischen Buchstaben z.B:

**Wort**

Ein Wort w der Länge n über dem Alphabet Σ ist ein n-Tupel, w ∈ Σn.

**Leeres Wort**

Das leere Wort ist das Wort mit der Länge 0 es wird mit ε bezeichnet.

**Sprache**

Eine Sprache L über dem Alphabet Σ ist eine Teilmenge von Σ∗, also L ⊂ Σ∗. Über jedem Alphabet gibt es die leere Sprache ∅ und die Sprache, die nur aus dem leeren Wort

besteht {ε}

**Regulär**

Den Regeln, Bestimmungen, Vorschriften entsprechend

**DEA**

**D**eterministischer **E**ndlicher **A**utomat: Gibt man nun eine Eingabe, wobei nur Zeichen enthalten sein können, die im Eingabealphabet stehen, in den Automaten ein, dann passiert für jede Eingabe ein Zustandsübergang. Hierbei gilt, dass ein deterministischer endlicher Automat immer eindeutig ist, bei welcher Eingabe welcher Zustandsübergang ausgeführt wird.

**nicht deterministisch**

Nichtdeterminismus ist ein Konzept aus der theoretischen Informatik, in dem Algorithmen oder Maschinen nicht nur genau eine Berechnung zu einer bestimmten Eingabe durchlaufen können, sondern es bei gleicher Eingabe mehrere Möglichkeiten für den Übergang in den nachfolgenden Zustand gibt.

**NEA**

Ein nichtdeterministischer endlicher Automat ist ein endlicher Automat, bei dem es für den Zustandsübergang mehrere gleichwertige Möglichkeiten gibt.

**Akzeptieren**

Ein DEA oder NEA akzeptiert wen der jeweilige Automat durch ein Input Wort in den Akzeptierzustand überführt wurde.

**Akzeptierte Sprache eines DEA**

Die Akzeptierte Sprache eines DEA ist eine Reguläre Sprache

**Regulärer Ausdruck**

Ein regulärer Ausdruck ist in der theoretischen Informatik eine Zeichenkette, die der Beschreibung von Mengen von Zeichenketten mit Hilfe bestimmter syntaktischer Regeln dient

**Grammatik**

Formale Grammatiken sind mathematische Modelle von Grammatiken, die zur eindeutigen Erzeugung und Beschreibung formaler Sprachen dienen.

(Eine formale Sprache besteht aus einer bestimmten Menge von Symbolketten (im Allgemeinen Zeichenketten) („Wörter“ der Sprache), die aus einem Zeichen-/Symbolvorrat („Alphabet“, Grundsymbole) zusammengesetzt werden können.)

**Kontextfrei**

Der Name «kontextfrei» rührt daher, dass es bei der Anwendung der Regeln nicht auf den Kontext ankommt, in dem eine Variable auf der linken Seite des Pfeils vorkommt. Regeln der Form aA → AA, bA → BB sind kontextsensitive Regeln.

**Variable**

**Text

Description automatically generated with low confidence**in diesem Fall ist das A eine Variable.

**Regel**

Eine Regel in den Kontextfreien Sprachen beschreibt den übergang von einer Variablen zu einer Variablen/Variablenkete oder einem Terminalsymbol.

**Ableitung**

Die Ableitung im zeigt auf wie man von einer Startvariable zu einem gegebenen Wort kommt.

**Parse Tree**

Der Ableitungsbaum eines Wortes einer kontextfreien Sprache ist eine Darstellung der verwendeten Produktionsregeln in Baum-Struktur. Die Grammatik für arithmetische Ausdrücke produziert zum Beispiel den Ausdruck 7 \* (3 + 5). Die dabei verwendeten Regeln können in Baumform dargestellt werden.

**Erzeugte Sprache einer Grammatik**

Mithilfe einer Grammatik lassen sich die Wörter einer Sprache L erzeugen: Ein Wort gehört zu L, wenn es sich durch eine Folge von bestimmten zulässigen Ersetzungen aus einem sogenannten Startsymbol erzeugen lässt

**Stackautomat**

Ein Stackautomat ist ein Automat der über einen primitiven Speicher den sogenannten Stack verfügt auf welchem er Variablen lesen und schreiben kann. Der Speicher funktioniert nach dem Prinzip Last-in-First-out (Typischer Stack).

**Turing-Maschine**

Die Turingmaschine hat ein Steuerwerk, in dem sich das Programm befindet, und besteht außerdem aus

* einem unendlich langen Speicherband mit unendlich vielen sequentiell angeordneten Feldern. Pro Feld kann genau ein Zeichen aus einem vordefinierten Alphabet gespeichert werden. Als zusätzliches Zeichen ist ein Blank (englisch für „leer/unbeschrieben“) zugelassen, das einem leeren Feld auf dem Speicherband entspricht.
* einem programmgesteuerten Lese- und Schreibkopf, der sich auf dem Speicherband feldweise bewegen und die Zeichen verändern (im Fall eines zu ‚schreibenden‘ Blanks auch löschen) kann.

**Erkannte Sprache einer Turing-Maschiene**

Ein TM akzeptiert eine Sprache L wenn die Eingabe jedes Wortes x ∈ L nach endlich vielen Schritten in einem akzeptierenden Zustand hält und bei x ∉ L in einem nicht akzeptiert zustand oder überhaupt nicht hält.

**Entscheider**

Ein Entscheider ist eine Turing-Maschiene die auf jedem beliebigen Input anhält.

**Entscheidbare Sprache einer TM**

Turing-Maschine akzeptiert eine Sprache und **hält garantiert** in endlicher Zeit, auch bei x ∉ L.

**Akzeptanzproblem**

Das Akzeptanzproblem beschreibt die Frage, ob ein Automat einen Input akzeptiert oder nicht für DEAs und CFG ist das Akzeptanzproblem entscheidbar für eine TM nicht.

**Abzählbar unendlich**

In der Mengenlehre wird eine Menge A als abzählbar unendlich bezeichnet, wenn sie die gleiche Mächtigkeit hat wie die Menge der natürlichen Zahlen. Dies bedeutet, dass es eine Bijektion zwischen A und der Menge der natürlichen Zahlen gibt, die Elemente der Menge A also „durchnummeriert“ werden können.

**Überabzählbar**

Eine Menge heißt überabzählbar, wenn sie nicht abzählbar ist. Eine Menge ist also genau dann überabzählbar, wenn ihre Mächtigkeit (entspricht der Anzahl der Elemente bei endlichen Mengen) größer ist als die der Menge der natürlichen Zahlen.

**Polynomielle Laufzeit**

z.B O(nk**)**

**P und NP**

***Klasse P:*** Zu der Klasse P gehören Sprachen die mit einem Entscheider in

polynomieller Laufzeit (nk) entschieden werden können

(Entscheider = TM die garantiert anhält, also deterministische TM)

Bsp: Kontextfreie Sprachen, Teilerfremdheit

***Klasse NP:*** NP ist genau dann die Klasse der Sprachen, für die ein deterministischer polynomieller Verifizierer existiert. Zu der Klasse NP gehören Sprachen, die in polynomieller Zeit von einer nichtdeterministischen Turing Maschine entschieden werden können.

**NP-vollständig**

Die schwersten Probleme in NP 🡪 siehe Katalog von Karp

**Polynomielle Reduktion**

Bei der Polynomiellen Reduktion muss die Reduktionsabbildung auch noch polynomielles Laufzeitverhalten haben.

**Was ist Σ∗?**

Die Menge aller Wörter beliebiger Länge aus dem Alphabet Σ.

Beispiel:

Σ = {0, 1, . . . , 9}.

Σ\* ist die Menge aller Ziffernfolgen, also 0, 1, 11, 123 usw.

**Was ist der Unterschied zwischen ε, ∅ und {ε}?**

ε: das leere Wort

∅: die leere Sprache

{ε}: die Sprache die nur aus dem leeren Wort besteht

**Wie unterscheidet sich ein deterministischer endlicher Automat von einem nicht deterministischen endlichen Automaten?**

In einem DEA sind alle Übergänge von einem in den Anderen Zustand eindeutig wohingegen in einem NEA die Übergänge nicht eindeutig sind. Weiter kann ein NEA ε-Übergänge haben welche es in eine DEA nicht geben kann.

**Wie kann man zwei endliche Automaten vergleichen?**

In dem man von beiden Automaten den Minimal-Automaten bildet ist dieser Automat derselbe sind die Automaten gleich.

**Nennen Sie drei Methoden, mit denen Sie zeigen können, dass eine Sprache regulär ist.**

1. DEA konstruieren
2. NEA konstruieren
3. REGEX konstruieren

**Nennen Sie zwei Methoden, mit denen Sie zeigen können, dass eine Sprache nicht regulär ist.**

1. Pumping Lemma
2. Myhill-Nerode

**Beschreiben Sie den Zusammenhang zwischen DEAs, NEAs und regulären Ausdrücken.**

Ein NEA kann in einen DEA umgewandelt werden welcher wiederum in einen regulären Ausdruck umgewandelt werden kann.

**Beschreiben Sie DEAs für die Sprachen ∅, {ε} und Σ ∗.**

Σ\*: Komplettes Alphabet

A picture containing qr code

Description automatically generatedDiagram

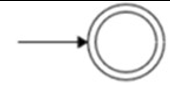
Description automatically generated

∅: die leere Sprache

A picture containing diagram

Description automatically generated

{ε}: die Sprache die nur aus dem leeren Wort besteht



**Warum sind endliche Sprachen regulär?**

Aus endlichen Sprachen lässt sich immer ein DEA konstruieren und ein DEA akzeptiert nur reguläre Sprachen.

**Geben Sie ein typisches Beispiel für eine nicht reguläre Sprache.**

L = {0n1n| n ≥ 0}

**Was ist eine kontextfreie Grammatik?**

Formale Grammatik die nur Ersetzungsregeln enthält, bei welcher immer genau nur ein Nichtterminalsymbol auf eine beliebig lange Folge von Nichtterminal - und Terminalsymbolen abgeleitet wird. S -> w Das Zeichen links vom Pfeil hängt nur davon ab, ob das Nichtterminalsymbol rechts in der Zeichenkette vorkommt. Es hängt aber nicht davon ab in welchem Kontext es sich befindet.

**Was bedeutet w ∈ L(G)?**

Das Wort w ist Element der Sprache L welche durch die Grammatik G erzeugt wurde.

**Welche Eigenschaften hat eine kontextfreie Grammatik in Chomsky-Normalform?**

* Man kann sofort abschätzen, wie viele Regelanwendungen notwendig sind, um ein gegebenes Wort zu erzeugen
* Startvariable ist nur auf der linken Seite
* Es gibt keine Regel A → ε
* Keine Unit-Rules
* Keine Verkettungen von Regeln

**Ist die Chomsky-Normalform eindeutig?**

Nein sie ist nicht eindeutig, da es mehrere Parse-Tree zum gleichen Wort haben kann.

**Welche Eigenschaften muss eine Sprache haben, damit es einen Stack-Automaten gibt, der sie akzeptieren kann?**

Die Sprache muss kontextfrei sein.

# kontrollfragen und antworten

**Geben Sie Grammatiken an für die Sprachen ∅, {ε} und Σ ∗ .**

∅: Keine Grammatik

{ε}: S -> ε

Σ\*: S -> ε

S -> SA

A -> a | b | c| …| z

**Geben Sie ein typisches Beispiel für eine nicht kontextfreie Sprache.**

L = {anbncn | n≥0}

**Gibt es eine Turing-Maschine mit nur einem Zustand?**

Nein eine Turing Maschine muss immer zwei Zustände haben qaccept und qreject

**Wie viele verschiedene Sprachen können von Turing-Maschinen mit zwei Zustanden erkannt werden. Warum?**

* Es können zwei Sprachen erkannt werden
* Eine Turing Maschine braucht immer zwei Zustände qaccept und qreject
* Die Sprachen sind Σ\* und die Leere Sprache ∅

**Zählen Sie drei Varianten von Turing-Maschinen auf.**

* Mehrspurige TM
* Aufzähler (TM mit drucker)
* Nicht deterministische TM

**Sei M eine nicht deterministische Turing-Maschine und w ∈ Σ∗. Was heisst w ∈ L(M)?**

M wird auf Input w garantiert anhalten.

**Was ist der Unterschied zwischen einer Turing erkennbaren Sprache und einer Turing entscheidbaren Sprache?**

* Turing erkennbar bedeutet das die TM bei jedem Wort w ∈ L nach endlich vielen Schritte in einem akzeptier Zustand anhält und bei x ∉ L in einem nicht akzeptier Zustand oder gar nicht hält
* Bei einer Touring entscheidbaren Sprache hält die TM garantiert in endlicher Zeit auch bei x ∉ L**.**

**Was müssen Sie tun, um nachzuweisen, dass eine Turing-erkennbare Sprache entscheidbar ist?**

* Entscheidbar heisst, man kann ein Programm schreiben, welches garantiert immer anhält.
* Das die Sprache erkennbar wird, müsste man das Programm verbessern, damit die Maschine auch immer anhält

**Beschreiben Sie das prototypische nicht entscheidbare Problem für Turing-Maschinen.**

* Das Prototypische Problem ist das Akzeptanzproblem (ATM)
* fragt, ob eine Turing Maschine ein Inputwort w erkennen wird

**Erklären Sie eine Standardtechnik, mit der man nachweisen kann, dass ein Problem nicht entscheidbar ist**.

Reduktion auf das Halteproblem oder Satz von Rice

**Geben Sie ein Beispiel eines nicht entscheidbaren Problems.**

* Akzeptanzproblem(Atm)
* fragt, ob eine TM ein Wort w akzeptieren wird
* Halteproblem(HALTtm)
* fragt, ob die Maschine M auf dem Wort w anhalten wird
* Leerheitsprobelm(Etm)
* fragt, ob die akzeptiere Sprache der Maschine M die Leere Sprache akzeptiert
* Regularitätsproblem(REGULARtm)
* frag, ob eine Maschine M erkennen kann, ob die akzeptiere Sprache der Maschine M regulär ist

**Was besagt das Halte-Theorem?**

Das Halte-Theorem besagt, dass es keine TM gibt welche für jegliche andere TM sagen kann, ob diese auf einem Inputwort w je anhalten oder endlos weiterlaufen werden

**Wie ändert sich die Laufzeit eines Algorithmus, wenn man von einer Variante einer TM zu einer Standard TM übergeht?**

* Eine Turingmaschine mit mehreren Bändern und Laufzeit t(n) kann in Laufzeit O(t(n)2) auf einer Standardturingmaschine simuliert werden.
* Nichtdeterministische TM's sind exponentiell schneller als deterministische TM's. Also die Laufzeit der nichtdeterministischen TM ist t(n) dann ist die Laufzeit der deterministischen 2O(t(n))

**Was bedeutet polynomielle Reduktion?**

* Man versucht ein Problem A auf ein anderes Problem B (welches mit polynomieller Laufzeit entscheidet) abzubilden
* Wenn ApB und B in Klasse P ist so ist auch A in Klasse P

**Wie können Sie herausfinden, ob eine Programmiersprache Turing-vollständig ist?**

Wen man mit einer Programmiersprache eine Turing Maschine simulieren kann ist die Sprache Turing-vollständig

**Warum ist die Sprache LOOP nicht Turingvollstandig?**

* Eine nicht Turing-vollständige Sprache, da sie immer terminiert und kein Halte-Problem besitzt
* Führt Schleifen aus, mit einer nicht veränderbaren Anzahl Durchführungen
* Schlüsselwörter LOOP, DO, END

**Gibt es Probleme, die man WHILE losen kann, nicht aber mit GOTO?**

* Nein GOTO und WHILE sind äquivalent
* Beide Sprachen sind Turing-vollständig

**Ist es moglich, einen Compiler zu schreiben, der C-Code in Brainfuck übersetzt?**

Ja es ist möglich hat sogar schonmal jemand gemacht. (Beide Sprachen sind Turing-vollständig)

# beispiele

## Logik

Zeigen Sie, dass die logischen Formeln und ( äquivalent sind

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | ( |
| t | t | t | t | t | t | t |
| t | t | f | t | t | t | t |
| t | f | t | t | t | f | f |
| t | f | f | t | t | t | t |
| f | t | t | t | t | t | t |
| f | t | f | t | f | t | f |
| f | f | t | f | t | f | f |
| f | f | f | f | f | t | f |

Für jedes rote t muss es möglich sein, mit Hilfe eines geeigneten Wahrheitswertes für in der Spalte ganz rechts ebenfalls ein rotes t zu erzeugen. Und wenn nicht wahr ist, wie in der letzten Zeile der Tabelle, dann darf auch kein Wahrheitswert für den Ausdruck

wahr machen.

## minimaler automat

Diagram

Description automatically generatedBetrachten Sie den folgenden, in Tabellenform gegebenen Automaten über das Alphabet

:

Table

Description automatically generated

Im ersten Schritt Paare aus Akzeptierzustand und Nicht-Akzeptierzustand markieren:

Table

Description automatically generated

A picture containing calendar

Description automatically generatedJedes noch nicht markierte Paar von Zuständen überprüfen, ob sie sich durch Anwendung der Übergangsfunktion in ein Paar überführen lässt, welches bereits markiert ist. Markieren, wenn dies der Fall ist. Diesen Schritt mehrfach iterieren.

Table

Description automatically generated with medium confidence

Durch Zusammenlegen im Zustandsdiagramm erhält man den kleinmöglichsten Automaten:

Diagram

Description automatically generated

## vollständige induktion

Text, letter

Description automatically generated

Text

Description automatically generated

## pumping lemma

Ist es möglich, einen regulären Ausdruck anzugeben, der auf Zeichenketten der folgenden Form passt? Eine solche Zeichenkette soll mit einer Anzahl n von Zeichengruppen 01 beginnen, gefolgt von genau n Zeichen 2.

Beweis nach Pumping Lemma:

1. Annahme, dass L regulär sei
2. Nach dem Pumping Lemma gibt es eine Pumping Length N, derart, dass jedes   
   Wort mit Länge aufgepumpt werden kann.
3. Wir konstruieren das Wort . Das Wort ist offensichtlich in L und
4. Nach den Annahmen des Pumping Lemma gibt es eine Aufteilung des Wortes in drei Teile , wobei . Dies zeigt, dass der Teil vollständig innerhalb des Teils aus 01 liegen muss.  
   A picture containing rectangle

   Description automatically generated
5. Beim Aufpumpten des Teils ändert sich die Zahl der Zeichen 0 oder 1, aber die Anzahl der 2 ändert nicht. Ein gepumptes Wort kann also nicht mehr in L sein
6. Dieser Widerspruch zeigt, dass die Annahme im Schritt 1 falsch war, dass also L nicht regulär sein kann.

In der Analysis lernt man, dass zu jedem Integralzeichen ein Symbol für die Integrationsvariable vorhanden sein muss. In diesem Verständnis ist

ein korrekter Integralausdruck,

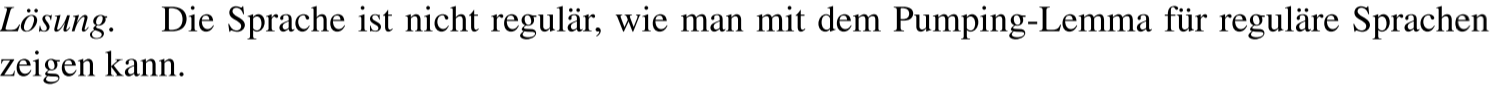
jedoch nicht. Kann man mit einem regulären Ausdruck testen, ob ein Ausdruck ein korrekter Integralausdruck ist?

Text, letter

Description automatically generated

Text

Description automatically generated



Graphical user interface, application, timeline

Description automatically generated

## pumping lemma für cfg

Können Sie eine Grammatik für die Sprache angeben?

Text, letter

Description automatically generated



Ist die Sprache kontextfrei?

Timeline

Description automatically generated

## polynomieller verifizierer

Das Logik-Rätsel Dedalo wird auf einem n × n-Feld gespielt. Darauf sind die Zahlen 1, . . . , k jeweils genau zweimal eingetragen, ein Feld kann aber höchstens eine Zahl enthalten. Gleiche Zahlen sollen jetzt durch einen Weg verbunden werden, wobei jedes Feld genau einmal von genau einem Weg getroffen werden darf, Wege können sich also nicht kreuzen. Es ist muss jedes Feld von einem Weg besucht sein, es ist nicht zulässig, dass ein Feld leer bleibt. Hier ist ein Dedalo-Rätsel mit eingezeichneter Lösung:

Diagram, engineering drawing

Description automatically generated

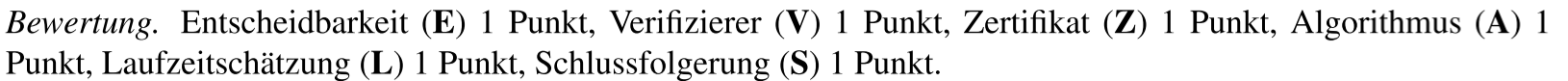
Zeigen Sie, dass eine nichtdeterministische Turing-Maschine in polynomieller Zeit entscheiden kann, ob ein Dedalo-Rätsel lösbar ist.

**Lösung:**

Die Frage ist sicher entscheidbar, indem man alle möglichen Wege durchprobiert und prüft, ob sie die verlangten Bedingungen erfüllen. Um die gestellte Frage zu beantworten, müssen wir einen polynomiellen Verifzierer konstruieren. Als Lösungszertifikat verlangen wir die Wege. Der Verifikationsalgorithmus muss folgendes überprüfen:

|  |  |  |
| --- | --- | --- |
| Schritt | Algorithmus | Laufzeit |
| 1 | Für jedes Feld kontrollieren, dass höchstens ein Weg dort durchgeht |  |
| 2 | Für jeden Weg kontrollieren, dass die Felder an den Enden die gleiche Nummer haben |  |
| 3 | Für jedes Feld kontrollieren, dass ein Weg dort durchgeht |  |
|  | Total |  |

Dieser Verifizierer hat offensichtlich polynomielle Laufzeit und daher kann eine nichtdeterministische Turing-Maschine in polynomieller Zeit entscheiden, ob ein Dedalo-Rätsel lösbar ist.



Nurikabe ist ein japanisches Logikrätsel, welches auf einem rechteckigen Gitter mit u × v Zellen gespielt wird. In einigen Feldern dieses Gitters sind Zahlen eingetragen. der Spieler muss einige der leeren Felder auswählen und schwarz färben. Die gefärbten Felder heissen “Fluss”. Es entstehen so weisse Gebiete, bestehend aus weissen Feldern, die entlang einer Kante benachbart sind. Es müssen folgende Regeln beachtet werden:

1. In jedem weissen Gebiet gibt es genau ein Zahlfeld, und die Zahl darin gibt an, wieviele weisse Felder das Gebiet umfasst
2. Es darf nur einen Fluss geben (das schwarze Gebiet muss zusammenhängend sein), und es darf keine schwarzen 2 × 2-Gebiete geben (der Fluss hat keine Tümpel)

Ist die Sprache in NP?

A screenshot of a game

Description automatically generated with low confidence

Table

Description automatically generated with medium confidence

Das Rätselspiel Fillomino wird auf einem n×m Spielfeld gespielt. In einzelnen Zellen des Spielfeldes sind Zahlen eingetragen. Der Spieler muss die leeren Felder mit Zahlen füllen, so dass zusammenhängende Gebiete soviele Felder enthalten wie die Zahl angibt. Ein zusammenhängendes Gebiet besteht aus Feldern, die sich entlang einer Kante berühren und alle die gleiche Zahl enthalten. Die Abbildung zeigt ein Fillomino-Rätsel (links) mit Lösung (rechts)

A screenshot of a game

Description automatically generated with medium confidence

Zeigen sie, dass eine nichtdeterministische Maschine in polynomieller Zeit entscheiden kann, ob ein Fillomino eine Lösung hat.

Text, letter

Description automatically generated

Text

Description automatically generated

Square Killer ist eine Variante von Sudoku, die auf einem n2 × n2 Spielfeld gespielt wird. Die normalen Sudoku-Regeln gelten, zusätzlich muss aber die Zahl der Ziffern in einem zusammenhängenden grauen Gebiet (“Käfig”) eine Quadratzahl sein. Das besondere an dieser Variante ist, dass es Square Killer-Rätsel gibt, die ganz ohne Vorgabe-Zahlen

auskommen, zum Beispiel:

Text, table

Description automatically generated

Kann eine nichtdeterministische Turing-Maschine in polynomieller Zeit entscheiden, ob ein Square Killer-Rätsel eine Lösung hat?

Text

Description automatically generated

## grammatik in chomsky normalform

Geben Sie eine Grammatik in Chomsky-Normalform für die Sprache Brainfuck.

Table

Description automatically generated

Die Sprache Brainfuck hat nur ein einziges Konstrukt, bei welchem auf die korrekte Schachtelung zu achten ist, nämlich die Schleifenkonstruktion mit [ und ]. Alle anderen Zeichen >, <, +, -, . und , können beliebig verwendet werden. Daraus können wir folgende

Grammatik ableiten:

A picture containing text, antenna

Description automatically generated

Schritte:

1. Neue Startvariable erzeugen  
   A picture containing antenna

   Description automatically generated
2. -Regeln entfernen  
   A picture containing text, clock, antenna

   Description automatically generated

A picture containing text, clock

Description automatically generated

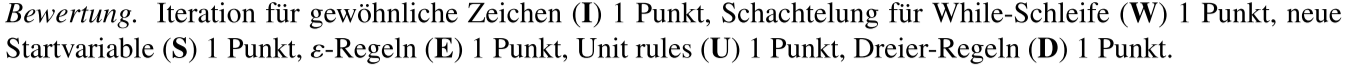
1. Text

   Description automatically generated with medium confidenceUnit-Rules entfernen und danach entfernen  
   Text

   Description automatically generated
2. Regeln mit rechter Seite [P] in akzeptable Form bringen  
   Text

   Description automatically generated  
   Text

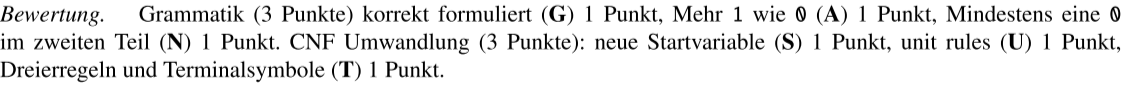
   Description automatically generated with low confidence



Ist die Sprache kontextfrei? Wenn ja, geben Sie CNF an.

Text, table

Description automatically generated



Ist die Sprache kontextfrei? Wenn ja, geben Sie CNF an.

Table

Description automatically generated

Table

Description automatically generated

## Halteproblem

Endlosschleifen gehören zu den immer wieder vorkommenden Programmierfehlern. Da ein JavaCompiler den vollständigen Überblick darüber hat, was in einem Java-Programm passiert, könnte man versuchen, die Änderungen, die der Code an den Variablen vornimmt, zu verfolgen. Dadurch könnte man dem Compiler ein Feature hinzufügen, mit dem der Compiler mögliche Endlosschleifen erkennen kann. Ist dies möglich?

**Lösung:**

Ein solcher Compiler wäre in der Lage, das Halteproblem für alle Programme zu lösen, die in dieser Sprache verfasst sind. Da Java Turing-vollständig ist, wäre damit das Halteproblem gelöst. Dies ist ein Widerspruch zum Haltetheorem.

## 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 , zu erzeugen, und die Zeichen in einem Wort sollen alphabetisch aufsteigend sortiert sein.

