DEA

Formale Definition eines DEA:

$$A = (Q, \Sigma, \delta, q_0, F)$$

- Q: Endliche Menge von Zuständen
- Σ: Alphabet
- δ: $Q \times \Sigma \to Q$ Übergangsfunktion
- $q_0 \in Q$: Startzustand
- $F \subset Q$: Menge der Akzeptierzustände

NEA

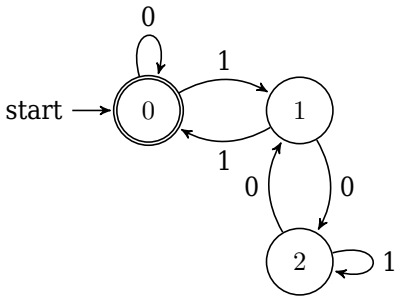
$$A = (Q, \Sigma, \delta, q_0, F)$$

- Q: Endliche Menge von Zuständen
- Σ: Alphabet
- δ: $Q \times (\Sigma \cup \{\varepsilon\}) \to P(Q)$ Übergangsfunktion
- $q_0 \in Q$: Startzustand
- $F \subset Q$: Menge der Akzeptierzustände

DEA minimieren

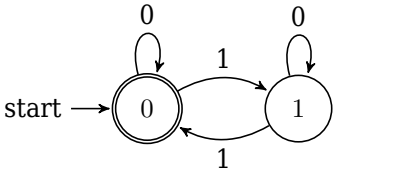
1. Tabelle aller Zustandspaare $\{q, q'\}, q \neq q'$ erstellen
2. Markiere alle Paare: $\{q, q'\}$ mit $q \in F$ und $q' \notin F$
3. Wiederhole folgende Schritte solange neue Markierungen hinzukommen
   3.1. Für jedes unmarkierte Paar $\{q, q'\}$ und jedes Zeichen $a \in \Sigma$:
   Wenn $\{\delta(q, a), \delta(q', a)\}$ markiert ist, dann markiere $\{q, q'\}$
4. Verschmelze unmarkierte Zustände

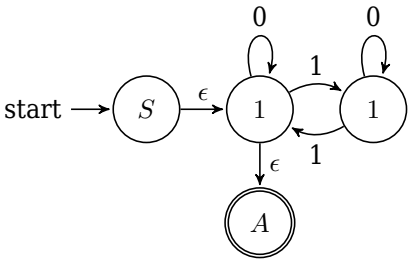
DEA: Durch 3 Teilbare Zahlen



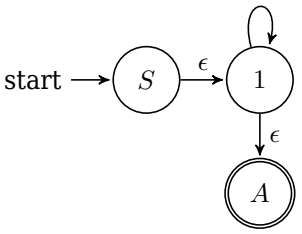
NEA/DEA zu Regulärem Ausdruck

1. Neue Start- und Akzeptierzustände
2. Zustände nacheinander entfernen





$0 \mid 10^*1$



Der Reguläre Ausdruck lautet: $(0 \mid 10^*1)^*$

L_n Sprachen Die Sprachen welche folgend Definiert werden sind regulär:

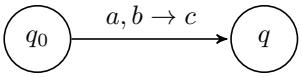
$$\begin{aligned} L_{ik} &= \{w \in \Sigma^* \mid |w|_i \cong k \mod n\} \\ &= L(i^k(i^n)^*) \end{aligned}$$

Die Sprache $L_{0k} \cap L_{1k}$ besteht aus Wörtern, in denen die Anzahl der 0 und den 1 den Rest k bei Teilung durch n haben. Als Durchschnitt regulärer Sprachen ist diese auch regulär. In der Sprache L_n sind alle solche Wörter mit den Resten $k = 0, \ldots, n-1$:

$$L_n = \bigcup_{k=0}^{n-1} L_{0k} \cap L_{1k}$$

Stack-Automat

Ein Stack-Automat verwendet ein Stack als Errinerung um CFG erkennen zu können.



Bedeutet, von Zustand q_0 geht es in Zustand q wenn der Input a enthält und auf dem Stack b zuoberst. Dabei wird c auf den Stack gelegt.

Kontextfreie Sprache

$$L = (V, \Sigma, R, S)$$

- V: Endliche Menge von Variablen
- Σ: Endliche Menge von Zeichen (disjunkt zu V), Terminalsymbole.
- R Menge von Regeln
- $S \in V$: Startvariable

Grammatik formulieren Beispiel von Grammatiken mit Terminalsymoblen $\Sigma = \{0, 1\}$:

- Grammatik die jedes Wort enthält: $L = \Sigma^*$

$$W \to W\mathtt{0} \mid W\mathtt{1} \mid \varepsilon$$

- Wörter, die mit dm gleichen Symbol enden wie sie beginnen:

$$\begin{aligned} E &\to \varepsilon \mid \mathtt{0} \mid \mathtt{1} \mid \mathtt{0}W\mathtt{0} \mid \mathtt{1}W\mathtt{1} \\ W &\to W\mathtt{0} \mid W\mathtt{1} \mid \varepsilon \end{aligned}$$

- L enthält alle Wörter gerader Länge:

$$\begin{aligned} G &\to GP \mid \varepsilon \\ P &\to ZZ \\ Z &\to \mathtt{0} \mid \mathtt{1} \end{aligned}$$

- $L = \{0^n 1^m | n > m\}$

$$S \to \mathtt{0}S \mid \mathtt{0}S\mathtt{1} \mid \mathtt{0}$$

- Erkennen von n-Tuple Addition:

$$\begin{aligned} S &\to \text{'('}\ E\ \text{')'} \\ E &\to N\ \text{','}\ E\ \text{','}\ N \\ &\to N\ \text{')'}\ \text{'+'}\ \text{'('}\ N \\ N &\to \ldots \end{aligned}$$

- Mindestens so viele o wie s (`o`, `oo`, `os`, `oos`, `oso`)

$$S \to \varepsilon \mid SS \mid oSs \mid oS$$

- Ausdrücke in korrekten Klammern

$$\begin{aligned} \text{min} &\to \text{number} \\ &\to \text{min} \sqcup \text{min} \\ &\to \text{kl.ausdruck} \\ \text{max} &\to \text{number} \\ &\to \text{max} \sqcap \text{max} \\ &\to \text{kl.ausdruck} \\ \text{kl.ausdruck} &\to \text{(min)} \\ &\to \text{(max)} \\ \text{number} &\to \text{digit} \\ &\to \text{number digit} \end{aligned}$$

- Vereinfachte LISP Grammatik

$$\begin{aligned} \text{liste} &\to \text{'('}\ \text{lst.elem.}\ \text{')'} \\ \text{lst.elem.} &\to \varepsilon \mid \text{lst.elem element} \\ \text{element} &\to \text{liste} \mid \text{atom} \\ \text{atom} &\to \text{zahl} \mid \text{string} \mid \text{symbol} \\ \text{zahl} &\to \text{ziffer} \mid \text{zahl ziffer} \\ \text{symbol} &\to \text{buchstabe} \mid \text{symbol bchstb} \\ \text{string} &\to \text{''}\ \text{stringinhalt}\ \text{''} \\ \text{stringinhalt} &\to \varepsilon \mid \text{stringinhalt zeichen} \\ \text{zeichen} &\to \text{ziffer} \mid \text{buchstabe} \end{aligned}$$

Pumping Lemma

Reguläre Sprachen Die **Pumping Lenght** ist die minimale Länge welche ein Wort haben muss, dass es aufgepumpt werden kann. In einem Automaten muss das Wort eine Schleife mindestens einmal durchlaufen haben.

$$w = xyz$$
$$|y| > 0$$
$$|xy| \leq N$$
$$xy^k z \in L \quad \forall k \geq 0$$

Kontextfreie Sprachen

$$w = uvxyz$$
$$|vy| > 0$$
$$|vxy| \leq N$$
$$uv^k xy^k z \in L \text{ für alle } k \in \mathbb{N}$$

Chomsky-Normalform

Eine kontextfreie Grammatik ist in Chomsky Normalform, wenn jede Regel in folgender Form ist:

$$A \to BC$$
$$A \to a$$
$$S \to \varepsilon$$

Wobei gilt:

$$A \in V,\ a \in \Sigma$$
$$B, C \in V \setminus \{S\}$$

CNF herstellen Algorithmus um eine Sprache in CNF zu bringen mit $\Sigma = \{a, b\}$:

1. Startvariable hinzufügen:
2. ε-Übergänge entfernen:
3. Unit-Rules entfernen:
4. Verkettungen:

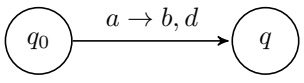
Turing Maschine

Formale Definition einer TM

$$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$$

- Q: Endliche Menge von Zuständen
- Σ: Endliche Menge des Input-Alphabets (ohne $\sqcup$)
- Γ: Endliche Menge des Bandalphabets (mit $\sqcup$), $\Sigma \subset \Gamma$
- δ: $Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ als Übergangsfunktion

Die Turing Maschine verwendet ein oder mehrere unendlich Lange Bänder um Sprachen erkennen zu können.



Von Zustand q_0 wird in Zustand q gewechselt, wenn unter dem Kopf ein a gelesen wird. Dabei wird das a mit einem b überschrieben und der Kopf in Richtung d bewegt.

Berechenbarkeit

Sei Σ ein festes Alphabet, dann sind folgende Mengen abzählbar unendlich:

1. Die Menge aller deterministischen endlichen Automaten.
2. Die Menge aller nichtdeterministischen endlichen Automaten.
3. Die Menge der regulären Sprachen.
4. Die Menge aller kontextfreien Grammatiken.
5. Die Menge aller kontextfreien Sprachen.
6. Die Menge aller Stackautomaten.
7. Die Menge aller Turingmaschinen.

Entscheidbarkeit

Eine Sprache ist **Turing-erkennbar** wenn es eine TM gibt, welche anhält und nur die Wörter der Sprache akzeptiert. Wörter welche nicht in der Sprache sind, werden von der TM entweder verworfen oder die *TM hält nie an*.
Eine Sprache ist **Turing-entscheidbar** wenn es eine TM gibt, welche Wörter in der Sprache akzeptiert und Wörter welche nicht in der Sprache sind, verwirft.

Entscheider Ein Entscheider ist eine Turingmaschine, die auf jedem Input $w \in \Sigma^*$ anhält. Eine Sprache heisst entscheidbar, wenn eine Entscheider sie erkennt.
Eine nicht deterministische Turingmaschine ist ein Entscheider, wenn jede mögliche Berechnungsgeschichte terminiert. Eine Sprache ist enscheidbar, wenn sie von einer nicht deterministischen Turingmaschine entschieden wird.

Verifizierer Ein Verifizierer für die Sprache A ist eine Turingmaschine V mit

$$A = \{w \mid \exists c \in C^* \text{ so dass } V\langle w, c\rangle \text{ akzeptiert}\}$$

wobei C eine endliche Menge ist. Ein Verifizierer heisst polynomiell, wenn seine Laufzeit polynomiell ist in der Länge des Wortes w. *Gibt es einen polynomiellen Verifizierer, so ist die Sprache in NP.*
Falls ein Verifizierer existiert, der das Problem in polynomieller Zeit verifizieren kann, kann das Problem auf einer nicht deterministischen Turingmaschine in polynomieller Zeit gelöst werden.

Akzeptanzproblem Gesucht ist ein Automat, der entscheidet ob ein anderer Automat ein Input akzeptiert.

$A_{\text{DEA}} = \{\langle B, w\rangle \mid$
B ist ein DEA und akzeptiert $w\}$
$A_{\text{NEA}} = \{\langle B, w\rangle \mid$
B ist ein NEA und akzeptiert $w\}$
$A_{\text{REX}} = \{\langle R, w\rangle \mid$
R ist ein REX und akzeptiert $w\}$
$A_{\text{CFG}} = \{\langle G, w\rangle \mid$ die CFG G erzeugt $w\}$
$A_{\text{TM}} = \{\langle M, w\rangle \mid$
M ist eine TM und erkennt $w\}$

DEA, NEA, REX (Regular Expression) und *CFG* sind entscheidbar, *TM* nicht.

Gleichheitsproblem Gesucht ist ein Automat der entscheidet, ob zwei andere Automaten die gleiche Sprache akzeptieren.

$EQ_{\text{DEA}} = \{\langle A, B\rangle \mid A, B$ sind DEAs und $L(A) = L(B)\}$
$EQ_{\text{CFG}} = \{\langle G, H\rangle \mid G, H$ sind CFGs und $L(G) = L(H)\}$
$EQ_{\text{TM}} = \{\langle M_1, M_2\rangle \mid M_i$ sind TMs und $L(M_1) = L(M_2)\}$

DEA ist entscheidbar, *CFG, TM* nicht.

Leerheitsproblem Gesucht ist ein Automat, der entscheidet, ob ein anderen Automat irgendein Wort akzeptiert.

$E_{\text{DEA}} = \{\langle A\rangle \mid$
A ist ein DEA und $L(A) = \emptyset\}$
$E_{\text{CFG}} = \{\langle G\rangle \mid$
G ist eine CFG und $L(G) = \emptyset\}$
$E_{\text{TM}} = \{\langle M\rangle \mid$
M ist eine TM und $L(M) = \emptyset\}$

DEA, CFG sind entscheidbar, *TM* nicht.

Halteproblem Das Halteproblem ist nicht entscheidbar

$\text{HALT}_{\text{TM}} = \{\langle M\rangle \mid$
M ist eine TM und hält auf $w\}$

Erzeugungsproblem Gesucht ist ein Automat der entscheidet ob eine CFG alle Wörter erzeugt.

$$ALL_{\text{CFG}} = \{\langle G\rangle \mid G \text{ ist CFG und } L(G) = \Sigma^*\}$$

Sprachprobleme

- Welche natürliche Zahlen sind Quadrate einer natürlichen Zahl

L eine Sprache über $\Sigma = \{0, 1\}$

$$L = \left\{ w \in \Sigma^* \;\middle|\; \begin{array}{l} w \text{ ist die Binär-} \\ \text{darstellung einer} \\ \text{Quadratzahl} \end{array} \right\}$$

- Falls $n \in \mathbb{N}$ eine Quadratzahl ist, finde man die Wurzel

L eine Sprache über $\Sigma = \{0, 1, :\}$

$$L = \left\{ w \in \Sigma^* \;\middle|\; \begin{array}{l} w \text{ ist von der} \\ \text{Form } w_1 : w_2, \\ \text{wobei } w_i \text{ Binar-} \\ \text{darstellungen} \\ \text{von Zahlen } n_i \\ \text{sind mit } n_1 = n_2^2. \end{array} \right\}$$

- Hat die quadratische Darstellung $ax^2 + bx + c = 0$ mit $a, b, c \in \mathbb{N}$ reele Lösungen

L eine Sprache über $\Sigma = \{0, 1, :\}$

$$L = \left\{ w \in \Sigma^* \;\middle|\; \begin{array}{l} w \text{ ist von der} \\ \text{Form } a : b : c, \text{ wo-} \\ \text{bei } a, b \text{ und } c \text{ Bi-} \\ \text{närdarstellungen} \\ \text{der Koeffizienten} \\ \text{einer quadrati-} \\ \text{schen Gleichung} \\ \text{sind, die reelle} \\ \text{Lösungen hat.} \end{array} \right\}$$

Diese Sprache kann mit einer TM entschieden werden, die die Diskriminante $b^2 - 4ac$ berechnet und im Zustand q_{accept} stehen bleibt genau dann, wenn die Diskriminante ≥ 0 ist.

- Hat die Gleichung $a^n + b^n = c^n$ ganzzahlige Lösungen, wobei mindestens eine der Zahlen a, b, c grösser als 1 sein muss.

L eine Sprache über $\Sigma = \{0, 1\}$

$$L = \left\{ w \in \Sigma^* \;\middle|\; \begin{array}{l} w \text{ ist die Bi-} \\ \text{närdarstellung} \\ \text{einer natürlichen} \\ \text{Zahl } n, \text{ für die} \\ \text{die Gleichung} \\ a^n + b^n = c^n \text{ eine} \\ \text{ganzzahlige Lö-} \\ \text{sung hat, wobei} \\ \text{mindestens eine} \\ \text{der Zahlen } a, b \\ \text{oder } c \text{ grösser als} \\ 1 \text{ sein muss.} \end{array} \right\}$$

- Man finde die Primfaktoren einer Zahl n

L eine Sprache über $\Sigma = \{0, 1, :\}$

$$L = \left\{ w \in \Sigma^* \;\middle|\; \begin{array}{l} w \text{ ist von} \\ \text{der Form} \\ n : p_1 : n_1 : \ldots : p_k : n_k, \\ \text{und es gilt} \\ n = p_1^{n_1} p_2^{n_2} \ldots p_k^{n_k}, \\ \text{wenn man die} \\ p_i \text{ und } n_i \text{ als} \\ \text{Binärzahlen} \\ \text{interpretiert.} \end{array} \right\}$$

- Goldbach-Vermutung: Jede Zahl > 2 könne als Summe von zwei Primzahlen geschrieben werden

L eine Sprache über $\Sigma = \{0, 1\}$

$$L = \left\{ w \in \Sigma^* \;\middle|\; \begin{array}{l} w \text{ ist die Binär-} \\ \text{codierung einer} \\ \text{einer geraden} \\ \text{Zahl } n, \text{ die als} \\ \text{Summe von zwei} \\ \text{Primzahlen ge-} \\ \text{schrieben werden} \\ \text{kann.} \end{array} \right\}$$

- Primzahlenzwillings-Vermutung: Es gibt unendlich viele Primzahlenzwillinge

L eine Sprache über $\Sigma = \{0, 1\}$

$$L = \left\{ w \in \Sigma^* \;\middle|\; \begin{array}{l} w \text{ ist die Binär-} \\ \text{codierung einer} \\ \text{Zahl } n, \text{ die eine} \\ \text{Primzahl ist so,} \\ \text{dass } n - 2 \text{ oder} \\ n + 2 \text{ ebenfalls} \\ \text{eine Primzahl ist.} \end{array} \right\}$$

NP-Probleme

SAT Satisfiability ist eine beliebig lange Sequenz aus Wahrheitswerten, zu denen eine Lösung gesucht wird.

$$\varphi = (x_1, \ldots, x_n)$$

Ein Lösungszertifikat wird die Belegung der Variablen verlangt und überprüft, ob φdurch einsetzten der Werte wahr wird.

3SAT Eine Variante von SAT ist 3SAT, welche nur Formeln in konjunktiver Normalsform vorkommen. Zudem hat jede Klausel genau 3 Therme.

$$\begin{aligned} \varphi = &(x_1 \vee x_2 \vee x_3) \\ &\wedge (\overline{x}_1 \vee \overline{x}_3 \vee \overline{x}_4) \\ &\wedge (x_1 \vee x_3 \vee x_5) \end{aligned}$$

k-Clique Eine k-Clique ist eine Menge von k Ecken des Graphen, so dass im Graph jede Ecke der Teilmenge mit jeder anderen Ecke verbunden ist. Das Problem ist in $O(n^k)$ lösbar. Als Lösungszertifikat verlangt man die Menge $c = \{v_1, \ldots, v_k\}$ der Vertizes der angeblichen Clique.

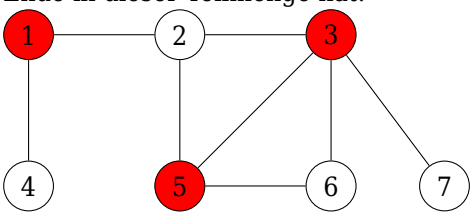
k-Vertex-Coloring Die Vertizes eines Graphen können mit k Farben eingefärbt werden, wenn sich für jeden Vertex eine der k Farben wählen lässt, so dass nie zwei durch eine Kante verbundene Vertizes die gleiche Farbe bekommen. Das Problem ist eintscheidbar (man kann alle n^k möglichen Färbungen ausprobieren). Als Lösungszertifkat verlant man die Farbzuordnung $c = (c_1, \ldots, c_n)$ der Ecken $1, \ldots, n$ des Graphen.

Hamiltonscher Pfad Ein Hamiltonscher Pfad in einem gerichteten Graphen ist ein Pfad, der jeden Vertex des Graphen genau einmal besucht.

Binary Integer Programming Für eine ganzzahlige Matrix C und einen ganzzahligen Vektor d soll ein binären Vektor x gefunden werden, sodass:

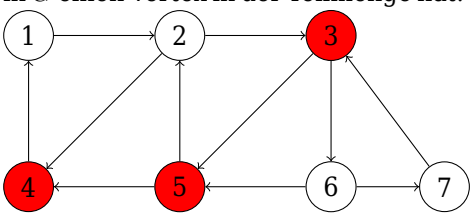
$$Cx = d$$

Vertex-Cover Ein Graph G und eine Zahl k. Gibt es eine Teilmenge von k Vertizes, sodass jede Kante von G ein Ende in dieser Teilmenge hat.



Telefonüberwachung: Knoten sind die Anschlüsse, die Kanten verbinden die Anschlüsse welche tatsächlich miteinander telefonieren. Gesucht ist die Menge aus$k$ Knoten, sodass jedes mögliche Gespräch abgehört werden kann, also jede Kante in einem Knoten dieser Menge endet.

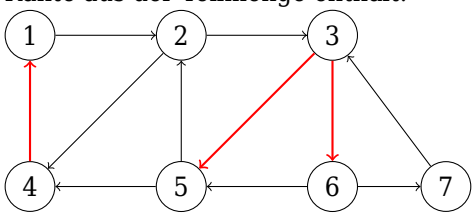
Feedback-Node-Set Gegeben ein gerichteter Graph G und eine Zahl k. Gibt es eine endliche Teilmenge von k Vertizes von G so, dass jeder Zyklus in G einen Vertex in der Teilmenge hat.



Roboter: Robter fahren Zyklen in einem gerichteten Graphen ab. Sie müssen durch regelmässig durch Prüfstellen fahren. Gesucht ist eine Menge von k Netzknoten sodass jeder Zyklus des Graphen durch eine dieser Prüfstellen verläuft.

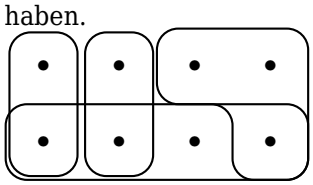
Feedback-Arch-Set Gegeben sei ein gerichteter Graph G und eine Zahl k. Gibt es eine Teilmenge aus k Kanten, sodass jeder Zyklus in G eine

Kante aus der Teilmenge enthält.



Roboter: Die Prüfstellen können die Roboter nun während er Fahrt prüfen. Gesucht ist eine Menge von k Kanten, sodass jeder Zyklus des Graphen über eine dieser Kanten führt.

Set-Covering Gegeben eine endliche Familie endlicher Mengen $(S_j)_{1\leq j\leq n}$ und eine Zahl k. Gibt es eine Unterfamilie bestehend aus k Mengen, welche dieselbe Vereinigung haben.

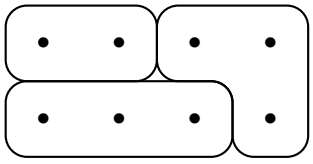


Steuererleichterungen Jeder Wähler profitiert mindestens von einer Steuererleichterung. Es muss die kleinste Menge von Vereinigungen gefunden werden, sodass die gesamte Menge in der Vereinigung abgebildet ist. Die Steuervergünstigungen sind nummeriert $1\dots n$. Sei S_j die Menge der Wähler, welche von Vergünstigung i profitieren. Es muss nun eine Teilmenge $I = \{i_1, i_2, \dots, i_k\}$ gefunden werden, dass die Vereinigungsmenge von Steuervergünstigungen der Wähler (S_j) und die Menge der Vergünstigung gleich sind. Also:

$$\bigcup_{i=1}^{n} S_i = \bigcup_{i\in I} S_i$$

Exact Cover Gegeben eine Familie $(S_j)_{1\leq j\leq n}$ von Teilmengen einer Menge U. Gibt es eine Unterfamilie von Mengen, die disjunkt sind, aber die gleiche Vereinigung haben. Die Unterfamilie $(S_{j_i})_{1\leq i\leq m}$ muss folgendes erfüllen:

$$S_{j_i} \cap S_{j_k} = \emptyset$$
$$\bigcup_{j=1}^{n} S_j = \bigcup_{i=1}^{m} S_{j_i}$$



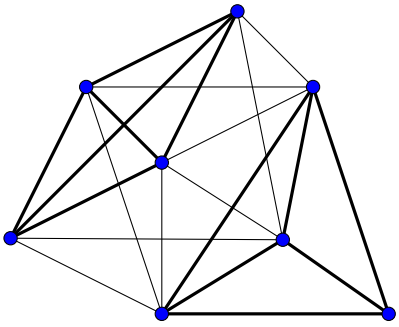
Sträflingslager: Die Menge aller Sträflinge ist U. Die Projekte sind nummeriert: $1\dots n$. Die Menge $S_i \subset U$ besteht aus den Sträflingen, welche für das Projekt i ungeeignet sind. Weil jeder Sträfling für mindestens ein Projekt ungeeignet ist, ist:

$$U = \bigcup_{i=1}^{n} S_i$$

Gesucht ist die Teilmenge von Projekten $I = \{i_1, \dots, i_m\} \subset \{1, \dots, n\}$ sodass jeder Sträfling genau auf einem Projekt arbeitet, aber kein Sträfling mehr als einem Projekt zugeteilt ist:

$$\bigcup_{j=1}^{m} S_{i_j} = U$$
$$S_{i_j} \cap S_{i_k} = \emptyset$$
$$\forall j \neq k$$

Clique-Cover Gegeben ist ein Graph und eine positive Zahl k. Gibt es k Anzahl Cliquen, sodass jede Ecke in genau einer Clique ist.

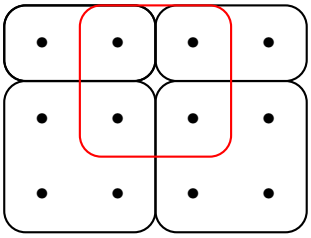


Gruppenarbeit: Für eine Gruppenarbeit sollen k Gruppen gebildet werden. Die Leute einer Gruppe sollen sich bereits gegenseitig kennen und alle Leute sollen beschäftigt sein.
Weak-Clique-Cover: Hier wird nicht verlangt, dass die Cliquen disjunkt sind.

3D-Matching Sei T eine endliche Menge und U eine Menge von Tripeln aus $T : U \subset T\times T\times T$. Gibt es eine Teilmenge $W \subset U$, sodass $|W| = |T|$ und keine zwei Elemente von W stimmen in irgendeiner Koordinate überein.
Marsbewohner: Auf dem Mars lebt eine Spezies mit drei Geschlechter. Von allen Geschlechtern gibt es genau n Personen. Es muss nun eine Liste aus Tripeln erstellt werden, um alle diese Personen in einer "Dreier"-Ehe verheiraten zu können.
Zwangsheirat: Es gibt n unverheiratete Frauen, Männer und Wohnungen. Es ist eine Liste mit einer Menge von Tripeln erstellt worden (x, y, z), wobei die Zahlen x, y und z aus der Menge $T = [n] = \{1, \dots, n\}$ stammen. Aus dieser Teilmenge $U \subset T \times T \times T$ soll jetzt eine Teilmenge $W \subset U$ von genau n Tripeln ausgewählt werden, sodass jedes Mann, jede Frau und jede Wohnung genau in einem Tripel vorkommt.

Hitting-Set Gegeben ist eine Menge von Teilmengen $S_i \subset S$. Gibt es eine Menge H die jede Menge genau in einem Punkt trifft:

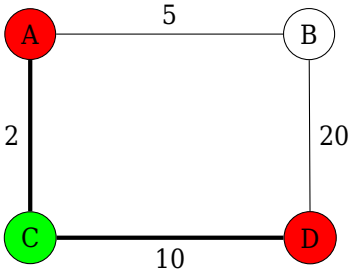
$$|H \cap S_i| = 1 \forall i$$



Fachgebiet: Sei S_i eine Menge von Wissenschaftlern, die zum Thema i Stellung beziehen können. Gesucht ist eine Auswahl H von Wissenschaftlern, sodass jeder Wissenschaftler genau ein Fachgebiet hat, für das er Stellung nehmen kann, also:

$$|H \cap S_i| = 1$$

Steiner-Tree Gegeben ist ein Graph, eine Teilmenge R von Vertizes und eine Gewichtsfunktion $w : E \mapsto \mathbb{Z}$ und eine positive Zahl k. Gibt es einen Baum mit Gewicht $\leq k$, dessen Knoten in R enhtalten sind. Das Gewicht des Baumes ist die Summer der Gewichte $w(\{u, v\})$ über alle Kanten $\{u, v\}$ im Baum.



Geldtransport: Bankgesellschaft möchte Filialen mit Geld versorgen. Geldtransporte sollen auch zwischen Filialen möglich sein. Vom Hauptsitz werden Beträge für mehrere Filialen verladen. Die Filialen nehmen die Lieferung ausseinander und laden sie auf neue Transporte. Die Frage ist, ob es möglich ist, mit dieser Methode die Kosten unter den Betrag k zu senken.

Sequencing Gegeben sei ein Vektor $(t_1, \dots, t_p) \in \mathbb{Z}^p$ von Laufzeiten von Jobs, ein Deadline Vektor $(d_1, \dots, d_p) \in \mathbb{Z}^p$, einem Strafenvektor $(s_1, \dots, s_p) \in \mathbb{Z}^p$ und eine positive Ganzzahl k. Gibt es eine Permutation π der Zahlen $1, \dots, p$ sodass die Gesamtstrafe für verspätete Ausführung bei der Ausführung der Jobs in der Reihenfolge $\pi(1), \dots, \pi(p)$ nicht grösser als k ist.

Job	Dauer	Deadline	Strafe
T1	4	5	30
T2	1	3	25
T3	2	2	20

Züge: Die Durchfahrzeit des Zuges i durch die Strecke ist t_i. Wenn der Zug später als d_i ankommt ist die Strafe s_i fällig. Wenn die Züge in der durch die Permutation π permutierten Reihenfolge $\pi(1), \pi(2), \ldots, \pi(n)$ abgefertigt, beträgt die für Zug j anfallende Strafe: $\vartheta(t_{\pi(1)}+\ldots+t_{\pi(j)})_{S_\pi(j)}$. Die Gesamtstrafe ist daher:

$$\sum_{j=1}^{n} \vartheta(t_{\pi(1)} + \ldots + t_{\pi(j)})_{S_\pi(j)}$$

Subset-Sum Gegeben ist eine Menge S von ganzen Zahlen. Kann darin eine Teilmenge gefunden werden, die als Summe einen bestimmten Wert t hat. **Jahresbudget:** Die Vorschläge des Teams, das Jahresbudget aufzubrauchen, bildet die Menge $\{b_1, \ldots, b_n\}$ von Beträgen b_i. Daraus soll eine Teilmenge $I = \{i_1, \ldots, i_m\}$ gebildet werden, sodass der Restbetrag r aufgebraucht wird.

$$\sum_{k=1}^{m} b_{i_k} = r$$

Partition Gegeben ist eine Folge von s ganzen Zahlen: $c_1, \ldots c_s$. Kann man die Indizes $1, \ldots, s$ in zwei Teilmengen I und $\bar{I}$ teilen, sodass die Summer der zugehörigen Zahlen identisch ist.

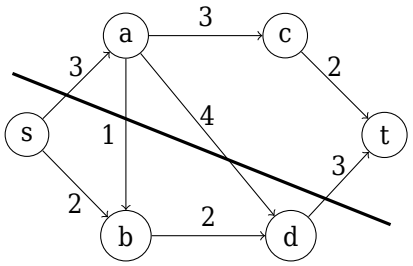
$$\sum_{i \in I} c_i = \sum_{i \notin I} c_i$$

Köngiserbe: Zwei Söhne eines verstorbenen Königs sollten genau denselben Betrag erben. Der Wert der Immobilie i ist mit c_i beziffert. Die Aufgabe besteht nun darin, eine Aufteilung zu der Menge A aller Immobilien zu finden, damit in die Summe der Mengen I und $A \setminus I$ genau diesselbe ist.

$$\sum_{i \in I} c_i = \sum_{i \in A \setminus I} c_i$$

Max-Cut Gegeben ist ein Graph mit einer Gewichtsfunktion $w : E \mapsto \mathbb{Z}$ und eine Ganzzahl 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:

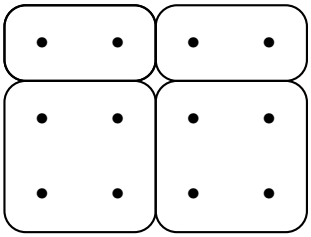
$$\sum_{\{u,v\} \in E \wedge u \in S \wedge v \notin S} w(\{u,v\}) \geq W$$



Grenzhandel: Die Produktionsstationen bilden einen Graphen, dessen Kanten anzeigen, ob zwischen den beiden Stationen ein Transport notwendig ist. Jede Kante ist der mögliche Gewinn zugeordnet, der winkt, wenn der Transport grenzquerend durchgeführt werden kann. Gesucht wird eine Aufteilung der Produktionsstationen auf die beiden Landesteile so dass der Gewinn aus Subventionen die Ziele W des Managements übersteigen.

Set-Packing Gegeben eine Familie $(S_i)_{i \in I}$ von Mengen und eine positive Ganzzahl k. Gibt es eine k-elementige Teilfamilie $(S_i)_{i \in J}$ mit $J \subset I$, dass die Mengen der Teilfamilie paarweise disjunkt sind:

$$\begin{aligned} |J| &= k \\ S_i \cap S_j &= \emptyset \\ \forall i, j \in J &\text{ mit } i \neq j \end{aligned}$$



Medizinische Studie: 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 $k = 17$ oder noch mehr ausgewählten Allergenen beschränken, die so beschaffen ist, dass kein Proband auf mehr als eines der ausgewählten Allergene reagiert.

Damen-Problem Das Acht-Damen-Problem ist die Schach-Aufgabe, acht Damen so auf einem Schachbrett aufzustellen, dass sie sich nicht gegenseitig schlagen können. Eine Dame kann eine andere Figur schlagen, die sich in der gleichen Zeile, Spalte oder Diagonale befindet. Offenbar ist acht die maximale Anzahl von Damen, die man auf diese Art auf einem Schachbrett platzieren kann.
Betrachten Sie jetzt das allgemeinere Problem

$$\mathit{DAMEN} = \left\{ n \;\middle|\; \begin{array}{l} \text{Es gibt eine Platzierung von } n \\ \text{Damen auf einem} \\ n \times n\text{-Schachbrett,} \\ \text{die sich nicht ge-} \\ \text{genseitig schlagen} \\ \text{können} \end{array} \right\}.$$

- Konstruieren Sie eine Reduktion von *DAMEN* auf *SAT*.
- Bestimmen Sie den Rechenaufwand Ihrer Konstruktion in Abhängigkeit von n.
- Wir müssen für jede Zahl n eine Formel φ_n in den Variablen x_{ij} mit $1 \leq i, j \leq n$ konstruieren, die genau dann erfüllbar ist, wenn sich n Damen auf dem Feld platzieren lassen, die sich nicht schlagen lassen.

  Diese Formel muss ausdrücken, dass in der gleichen Zeile, Spalte und Diagonalen keine weitere Dame steht. Wenn auf dem Feld (i, j) eine Dame steht, dann wird dies gemäss Hinweis dadurch ausgedrückt, dass x_{ij} wahr ist. Wir müssen daher eine Formel bauen, die sicherstellt, dass in diesem Fall alle x_{kl} falsch sind, die zur gleichen Zeile, Spalte oder Diagonalen gehören.

  Wir schreiben $\oplus$ für die XOR-Verknüpfung. Die Bedingung, dass keine weiteren Damen in der gleichen Zeile stehen, kann man durch die Formel

$$\begin{aligned} \varphi_{ij,\text{Zeile}} &= x_{ij} \oplus (x_{i1} \vee \cdots \vee \hat{x}_{ij} \vee \cdots \vee x_{in}) && (1) \\ &= x_{ij} \oplus \left(\bigvee_{k \neq j} x_{ik} \right) && (2) \end{aligned}$$

  ausdrücken. Dabei bedeutet $\hat{x}_{ij}$, dass die Variable x_{ij} in der Klammer auf der rechten Seite weggelassen werden soll. Analog kann man auch die Bedingungen für die Spalten und die Diagonalen ausdrücken:

$$\begin{aligned} \varphi_{ij,\text{Spalte}} &= x_{ij} \oplus (x_{1j} \vee \cdots \vee \hat{x}_{ij} \vee \cdots \vee x_{nj}) && (3) \\ &= x_{ij} \oplus \left(\bigvee_{l \neq i} x_{lj} \right), \\ \varphi_{ij,\text{Diagonalen}} &= x_{ij} \oplus \left(\bigvee_{k,\, l \text{ diagonal von } i,\, j} x_{kl} \right). \end{aligned}$$

  Die Variablen x repräsentieren genau dann eine akzeptable Platzierung wenn für jedes Paar (i, j) alle drei soeben entwickelten Formeln wahr werden:

$$\varphi_{ij} = \varphi_{ij,\text{Zeile}} \wedge \varphi_{ij,\text{Spalte}} \wedge \varphi_{ij,\text{Diagonalen}}.$$

  Die gesuchte Formel ist daher

$$\varphi = \bigwedge_{i,j=1}^{n} \varphi_{ij}.$$

- Für den Aufbau dieser Formel braucht für jedes Paar (i, j) den gleichen Aufwand $O(n)$. Es gibt n^2 solche Paare, der gesamte Aufwand ist daher $O(n^3)$.

Turing-Vollständigkeit

LOOP in RISC-LOOP: Allgemein übersetzen wir $x_j := x_i \pm c$ durch

- i `INCR` Befehle
- Zugriff $r := x_i$
- i `DECR` Befehle
- Rechnung $r := r \pm c$
- j `INCR` Befehle
- Speicherung $x_j := r$
- j `DECR` Befehle

Analog übersetzen wir `LOOP`x_i durch

- i `INCR` Befehle
- Zugriff $r := x_i$
- i `DECR` Befehle
- Schleifenbefehl `LOOP` r.

Auf diese Weise lässt sich jedes LOOP-Programm in eine RISC-LOOP Programm übersetzen, RISC-LOOP ist also mindestens so leistungsfähig wie LOOP. (LOOP ist jedoch nicht Turing-Vollständig)