Teil III
Komplexität

Vorlesung 13
Die Komplexitätsklassen P und NP

Whd.: Die Ackermann-Funktion

Definition

Die Ackermannfunktion $A\colon \mathbb{N}^2 \to \mathbb{N}$ ist folgendermaßen definert:

$$\begin{array}{lcll} A(0,n) & = & n+1 & \text{für } n \geq 0 \\ A(m+1,0) & = & A(m,1) & \text{für } m \geq 0 \\ A(m+1,n+1) & = & A(m,A(m+1,n)) & \text{für } m,n \geq 0 \end{array}$$

Wdh.: LOOP vs WHILE

Lemma

Für jedes LOOP-Programm P gibt es eine natürliche Zahl m, so dass für alle $n \in \mathbb{N}$ gilt: $F_P(n) < A(m,n)$.

Satz

Die Ackermannfunktion ist nicht LOOP-berechenbar.

Korollar

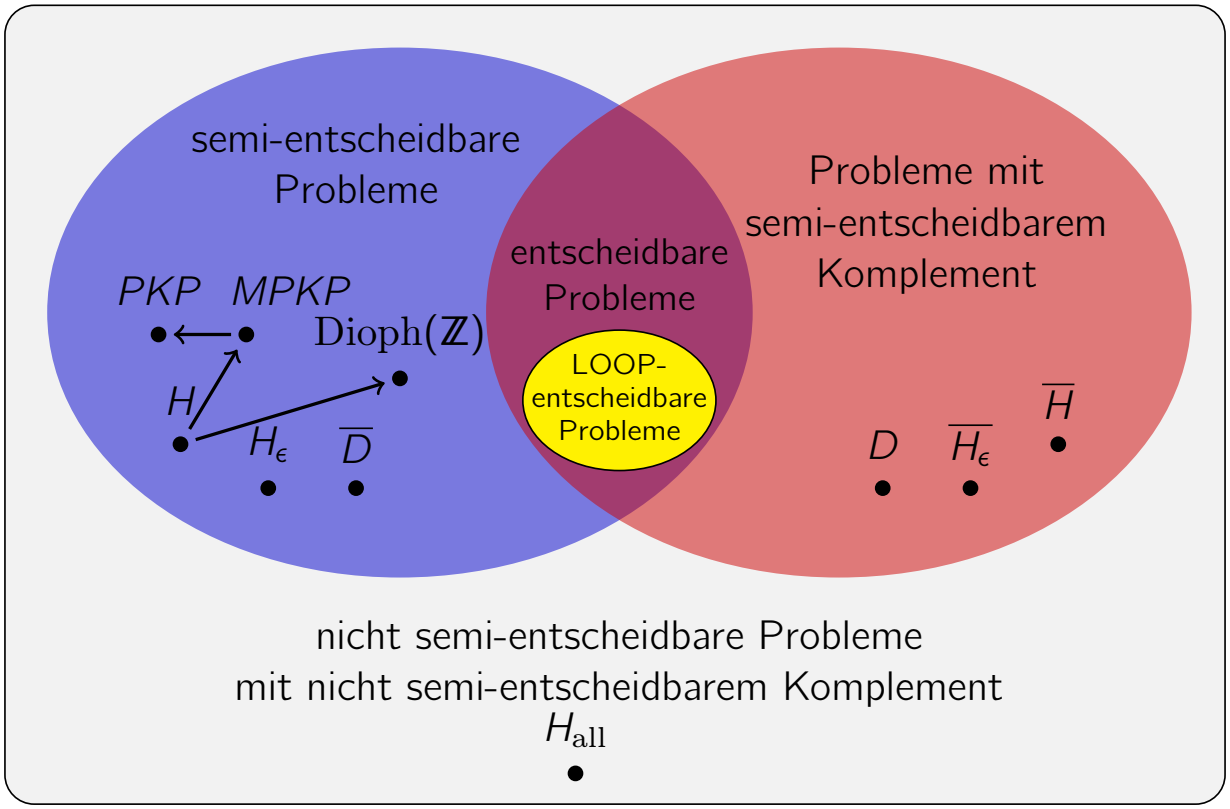
Die Klasse der LOOP-berechenbaren Funktionen ist eine echte Teilmenge der berechenbaren (totalen) Funktionen.

Bemerkung

Mit Hilfe eines Diagonalisierungsarguments lässt sich auch beweisen, dass es entscheidbare Sprachen gibt, die nicht LOOP-entscheidbar sind.

Wdh.: Berechenbarkeit

Berechenbarkeitslandschaft:



Wdh.: Kostenmodelle der RAM

Modelle für die Rechenzeit einer RAM

- Uniformes Kostenmaß: Jeder Schritt zählt eine Zeiteinheit.
- Logarithmisches Kostenmaß: Die Laufzeitkosten eines Schrittes sind proportional zur binären Länge der Zahlen in den angesprochenen Registern.

Definition von Polynomialzeitalgorithmus

Definition (worst case Laufzeit eines Algorithmus)

Die worst case Laufzeit $t_A(n)$, $n \in \mathbb{N}$, eines Algorithmus A entspricht den maximalen Laufzeitkosten auf Eingaben der Länge n bezüglich des logarithmischen Kostenmaßes der RAM.

Definition (Polynomialzeitalgorithmus)

Wir sagen, die worst case Laufzeit $t_A(n)$ eines Algorithmus A ist polynomiell beschränkt, falls gilt

$$\exists \alpha \in \mathbb{N} : t_A(n) = O(n^\alpha).$$

Einen Algorithmus mit polynomiell beschränkter worst case Laufzeit bezeichnen wir als Polynomialzeitalgorithmus.

Sortieren in Polynomialzeit

Problem (Sortieren)

Eingabe: N Zahlen $a_1, \ldots, a_N \in \mathbb{N}$
Ausgabe: aufsteigend sortierte Folge der Eingabezahlen

Anmerkung: Soweit wir nichts anderes sagen, nehmen wir an, dass Zahlen binär kodiert sind.

Sortieren in Polynomialzeit

Satz

Sortieren kann in Polynomialzeit gelöst werden.

Beweis:

- Wir lösen das Problem beispielsweise mit Mergesort.
- Laufzeit im uniformen Kostenmaß: $O(N \log N)$.
- Laufzeit im logarithmischen Kostenmaß: $O(\ell N \log N)$, wobei $\ell = \max_{1 \leq i \leq N} \log(a_i)$.
- Sei n die Eingabelänge. Es gilt $\ell \leq n$ und $\log N \leq N \leq n$.
- Somit ist die Laufzeit durch $\ell N \log N \leq n^3$ beschränkt.

□

Bemerkung: Sortieren ist kein Entscheidungsproblem.

Definition der Klasse P

Definition (Komplexitätsklasse P)

P ist die Klasse der Entscheidungsprobleme, für die es einen Polynomialzeitalgorithmus gibt.

Anmerkungen:

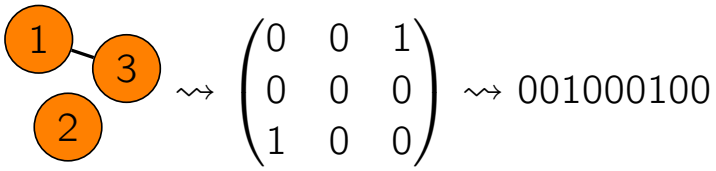
- Alternativ kann man sich auch auf die Laufzeit einer TM beziehen, da sich RAM und TM gegenseitig mit polynomiellem Zeitverlust simulieren können.
- Polynomialzeitalgorithmen werden häufig auch als „effiziente Algorithmen“ bezeichnet.
- P ist in diesem Sinne die Klasse derjenigen Probleme, die effizient gelöst werden können.

Graphzusammenhang in P

Problem (Graphzusammenhang)

Eingabe: Graph $G = (V, E)$
Frage: Ist G zusammenhängend?

Anmerkung: Bei Graphproblemen gehen wir grundsätzlich davon aus, dass der Graph in Form einer Adjazenzmatrix eingegeben wird.



Graphzusammenhang in P

Satz

Graphzusammenhang $\in$ P.

Beweis

- Wir lösen das Problem mit einer Tiefensuche.
- Laufzeit im uniformen Kostenmaß: $O(|V|^2)$,
  (sogar $O(|V| + |E|)$ bei Adjazenzlistenrepräsentation),
- Laufzeit im logarithmischen Kostenmaß: $O(|V|^2 \cdot \log |V|)$
- Die Eingabelänge ist $n = |V|^2$.
- Die Gesamtlaufzeit ist somit

$$O(|V|^2 \log |V|) \;=\; O(n \log n) \;=\; O(n^2) \;.$$

□

Weitere Beispiele für polynomiell lösbare Probleme

- Kürzester Weg
- Minimaler Spannbaum
- Maximaler Fluss
- Maximum Matching
- Lineare Programmierung
- Größter Gemeinsamer Teiler
- Primzahltest

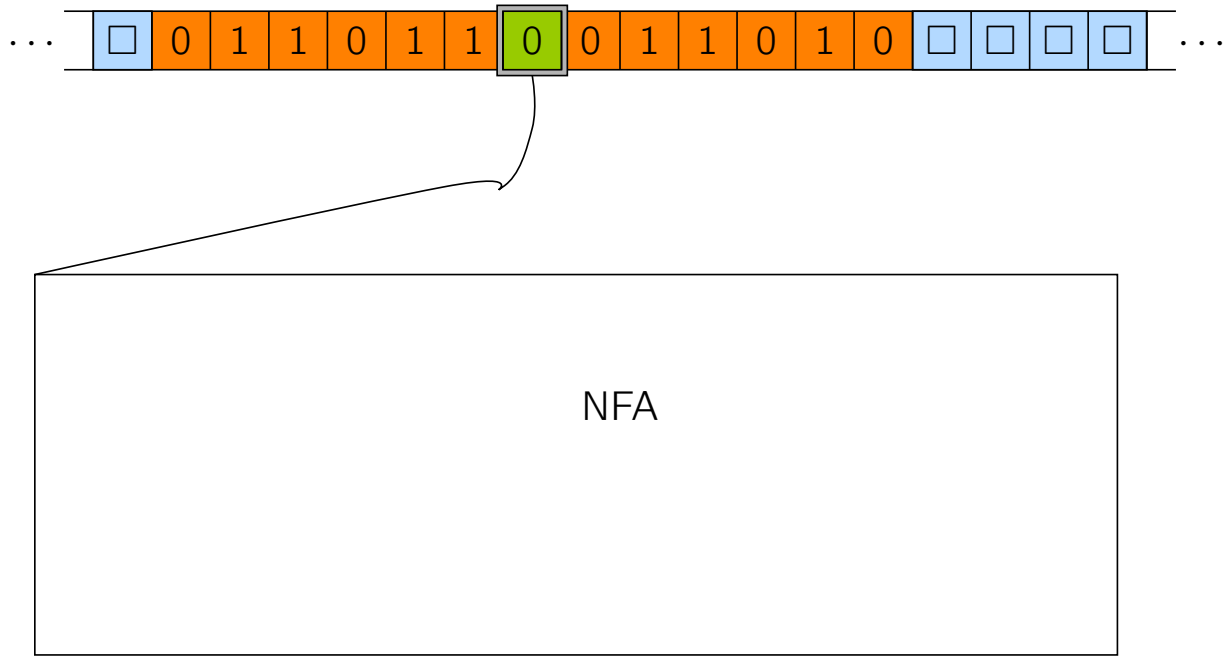
Definition von NTM

Definition (Nichtdeterministische Turingmaschine – NTM)

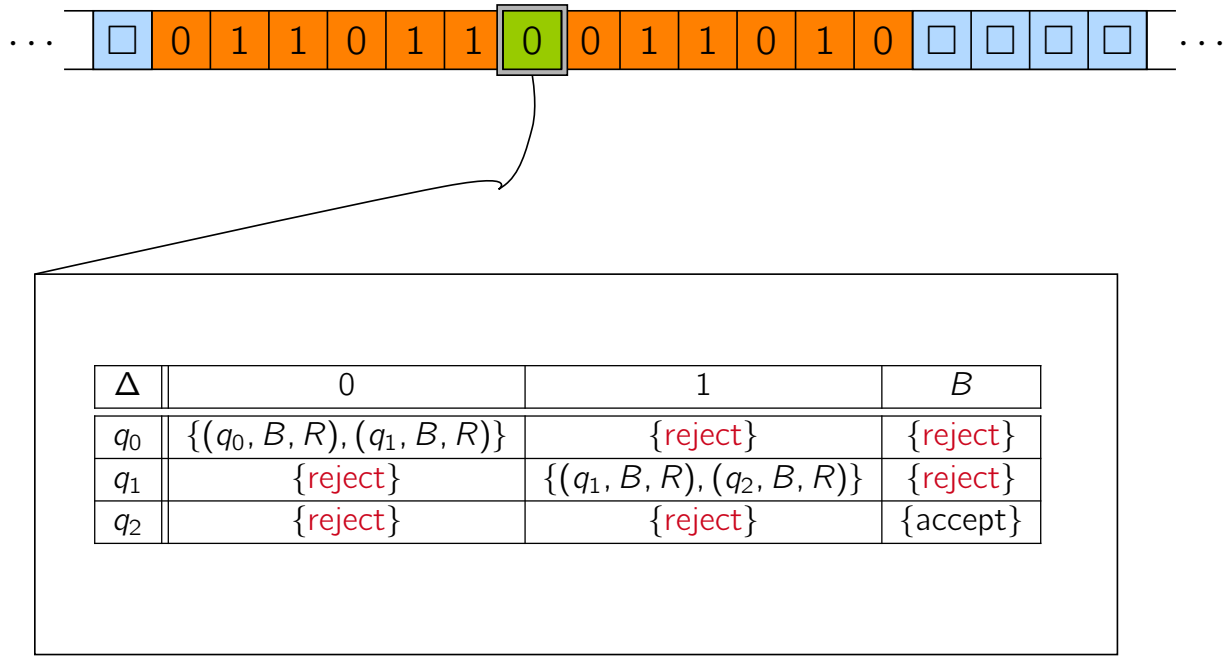
Eine nichtdeterministische Turingmaschine (NTM) ist definiert wie eine deterministische Turingmaschine (TM), nur die Zustandsübergangsfunktion wird zu einer Relation

$$\Delta \subseteq ((Q \setminus \{\bar{q}\}) \times \Gamma) \times (Q \times \Gamma \times \{L, R, N\}) \quad .$$

Nichtdeterministische Turingmaschine (NTM)



Nichtdeterministische Turingmaschine (NTM)



Δ	0	1	B
q_0	$\{(q_0, B, R), (q_1, B, R)\}$	{reject}	{reject}
q_1	{reject}	$\{(q_1, B, R), (q_2, B, R)\}$	{reject}
q_2	{reject}	{reject}	{accept}

Erläuterung der Rechnung einer NTM

- Eine Konfiguration K' ist direkter Nachfolger einer Konfiguration K, falls K' durch einen der in Δ beschriebenen Übergänge aus K hervorgeht.
- Rechenweg = Konfigurationsfolge, die mit Startkonfiguration beginnt und mit Nachfolgekonfigurationen fortgesetzt wird, bis eine Endkonfiguration im Zustand $\bar{q}$ erreicht wird.
- Der Verlauf der Rechnung ist also nicht deterministisch, d.h., zu einer Konfiguration kann es mehrere direkte Nachfolgekonfigurationen geben.

Erläuterung der Rechnung einer NTM

Die möglichen Rechenwege von M auf einer Eingabe $w \in \Sigma^*$ können in Form eines Berechnungsbaumes beschrieben werden:

- Die Knoten des Baumes entsprechen Konfigurationen.
- Die Wurzel des Baumes entspricht der Startkonfiguration.
- Die Kinder einer Konfiguration entsprechen den möglichen Nachfolgekonfigurationen.

Der maximale Verzweigungsgrad des Berechnungsbaumes ist

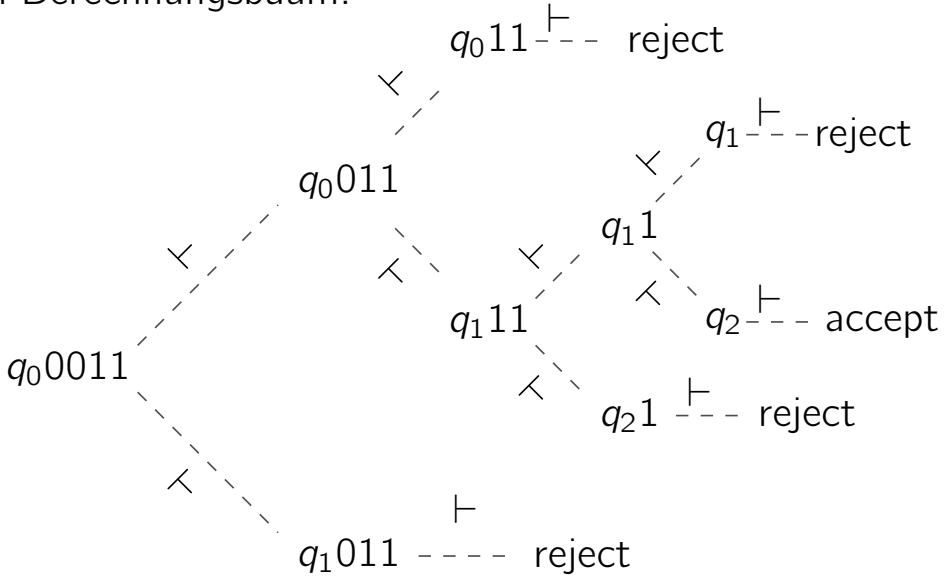
$$d = \max\left\{|\Delta(q,a)| \,\middle|\, q \in Q \setminus \{\bar{q}\}, a \in \Gamma\right\}.$$

Beachte, dass d nicht von der Eingabe abhängt, also konstant ist.

Berechnungsbaum - Beispiel

δ	0	1	B
q_0	$\{(q_0, B, R), (q_1, B, R)\}$	{reject}	{reject}
q_1	{reject}	$\{(q_1, B, R), (q_2, B, R)\}$	{reject}
q_2	{reject}	{reject}	{accept}

Die NTM mit dieser Übergangsrelation hat auf der Eingabe 0011 folgenden Berechnungsbaum:



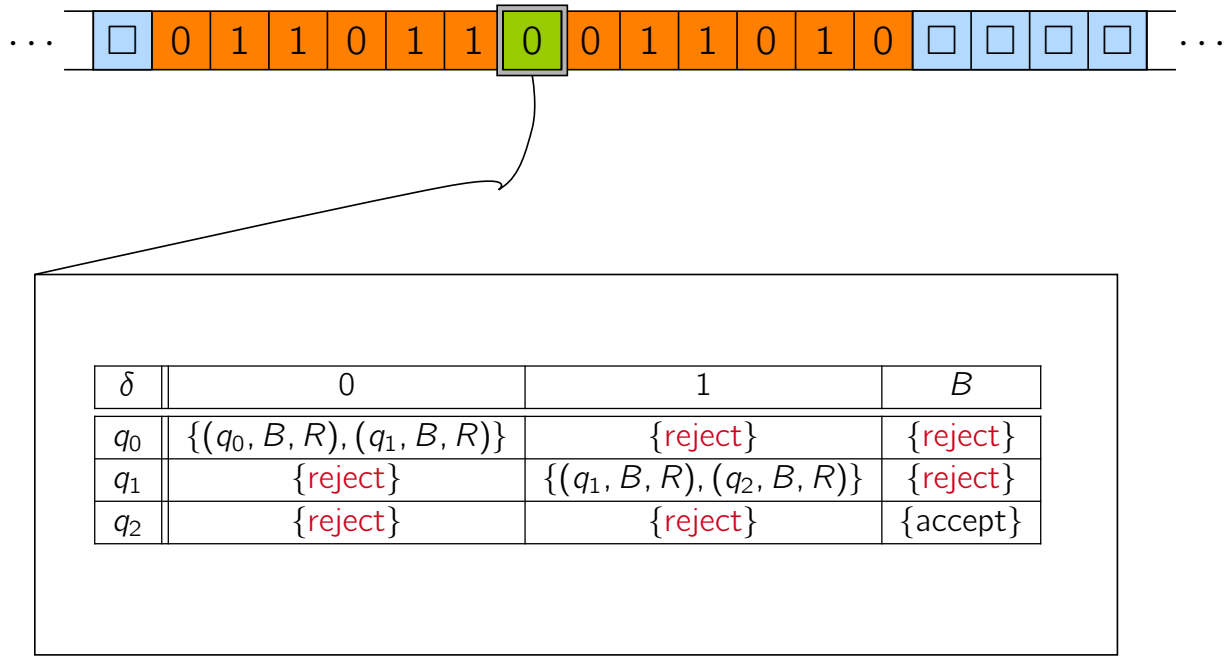
Definition des Akzeptanzverhaltens

Definition (Akzeptanzverhalten der NTM)

Eine NTM M akzeptiert die Eingabe $x \in \Sigma^*$, falls es mindestens einen Rechenweg von M gibt, der in eine Konfiguration mit akzeptierendem Zustand führt.

Die von M erkannte Sprache $L(M)$ besteht aus allen von M akzeptierten Wörtern.

Nichtdeterministische Turingmaschine (NTM)

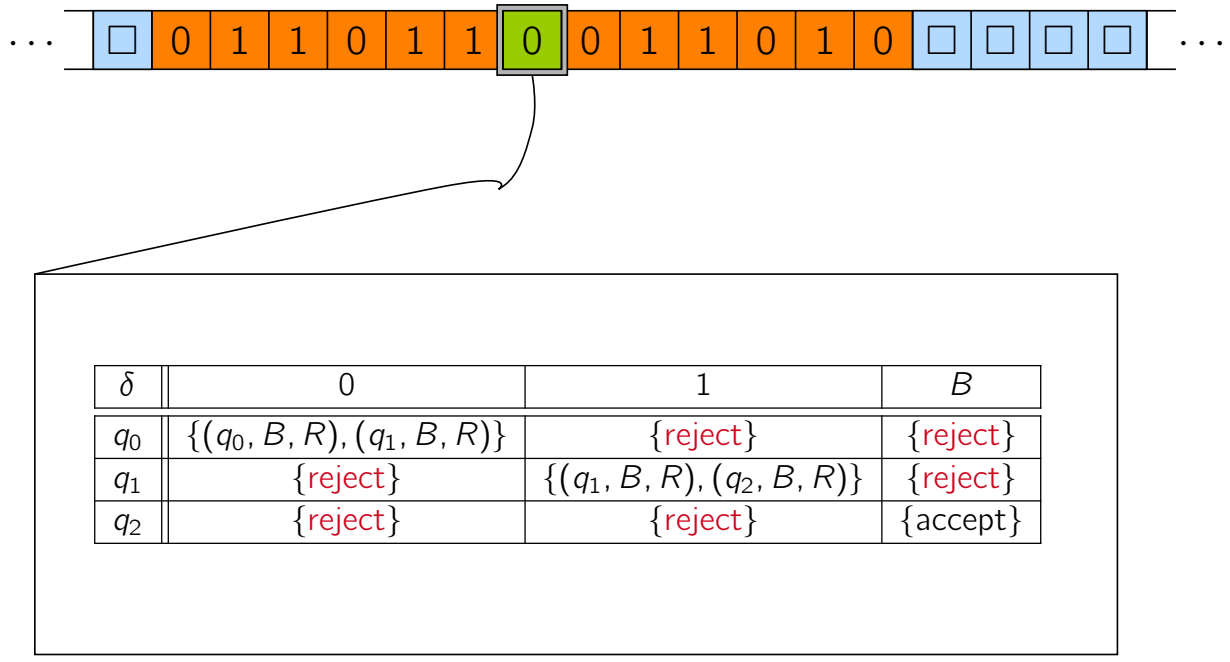


δ	0	1	B
q_0	$\{(q_0, B, R), (q_1, B, R)\}$	{reject}	{reject}
q_1	{reject}	$\{(q_1, B, R), (q_2, B, R)\}$	{reject}
q_2	{reject}	{reject}	{accept}

Welche Sprache wird von dieser NTM erkannt?

$$L(M) = \{0^i 1^j \mid i \geq 1, j \geq 1\}$$

Nichtdeterministische Turingmaschine (NTM)



δ	0	1	B
q_0	$\{(q_0, B, R), (q_1, B, R)\}$	{reject}	{reject}
q_1	{reject}	$\{(q_1, B, R), (q_2, B, R)\}$	{reject}
q_2	{reject}	{reject}	{accept}

Welche Sprache wird von dieser NTM erkannt?

$$L(M) = \{0^i 1^j \mid i \geq 1, j \geq 1\}$$

Definition der Laufzeit

Definition (Laufzeit der NTM)

Sei M eine NTM. Die Laufzeit von M auf einer Eingabe $x \in L(M)$ ist definiert als

$$T_M(x) := \text{Länge des kürzesten akzeptierenden Rechenweges von } M \text{ auf } x.$$

Für $x \notin L(M)$ definieren wir $T_M(x) = 0$.

Die worst case Laufzeit $t_M(n)$ für M auf Eingaben der Länge $n \in \mathbb{N}$ ist definiert als

$$t_M(n) := \max\{ T_M(x) \mid x \in \Sigma^n \}.$$

Definition der Klasse NP

Definition (Komplexitätsklasse NP)

NP ist die Klasse der Entscheidungsprobleme, die durch eine NTM M erkannt werden, deren worst case Laufzeit $t_M(n)$ polynomiell beschränkt ist.

NP steht dabei für nichtdeterministisch polynomiell.

Beispiel für ein Problem aus NP

Problem (Cliquenproblem – CLIQUE)

Eingabe: Graph $G = (V, E)$, $k \in \{1, \ldots, |V|\}$
Frage: Enthält G eine k-Clique?

- Für das Cliquenproblem ist kein Polynomialzeitalgorithmus bekannt.
- Die besten bekannten Algorithmen haben exponentielle Laufzeit.

Beispiel für ein Problem aus NP

Satz

$CLIQUE \in$ NP.

Beweis: Wir beschreiben eine NTM M mit $L(M) =$ CLIQUE:

1. Syntaktisch inkorrekte Eingaben werden verworfen.
2. M „rät“ einen 0-1-String y der Länge $|V|$.
3. M akzeptiert, falls
   - der String y genau k viele Einsen enthält und
   - die Knotenmenge $C = \{i \in V \mid y_i = 1\}$ eine Clique ist.

Korrektheit: Es gibt genau dann einen akzeptierenden Rechenweg, wenn G eine k-Clique enthält.

Laufzeit: Alle Schritte haben polynomielle Laufzeit. □

Die Komplexitätsklasse EXPTIME

Definition (Komplexitätsklasse EXPTIME)

EXPTIME ist die Klasse der Entscheidungsprobleme L, für die es ein Polynom q gibt, so dass sich L auf einer DTM mit Laufzeitschranke $2^{q(n)}$ berechnen lässt.

Wie verhält sich NP zu P und EXPTIME?

Wie verhält sich NP zu P und EXPTIME?

Wir setzen die Klassen P und EXPTIME mit der Klasse NP in Beziehung.

Satz

$\mathrm{P} \subseteq \mathrm{NP} \subseteq \mathrm{EXPTIME}$

Beweis:

Es gilt $\mathrm{P} \subseteq \mathrm{NP}$, weil eine DTM als eine spezielle NTM aufgefasst werden kann.

Wir müssen noch zeigen, dass $\mathrm{NP} \subseteq \mathrm{EXPTIME}$.

Exponentielle Laufzeitschranke für Probleme aus NP

Sei $L \in \mathrm{NP}$. Sei M eine NTM mit polynomiell beschränkter Laufzeitschranke $p(n)$, die L erkennt.

Sei $w \in \Sigma^*$. Wir konstruieren eine DTM M', welche die NTM M auf Eingabe w simuliert:

- In einer Breitensuche generiert M' den Berechnungsbaum von M bis zu einer Tiefe von $p(|w|)$.
- Falls dabei eine akzeptierende Konfiguration gefunden wird, so akzeptiert M' die Eingabe; sonst verwirft M' die Eingabe.

Exponentielle Laufzeitschranke für Probleme aus NP

Korrektheit:

- Falls $w \in L$ ist, so gibt es einen akzeptierenden Rechenweg von M der Länge $p(|w|)$. M' generiert diesen Weg und akzeptiert w.
- Falls $w \notin L$ ist, so gibt es keinen akzeptierenden Rechenweg von M der Länge $p(|w|)$. In diesem Fall wird w von M' verworfen.

Laufzeit:

Sei $d \geq 2$ der maximale Verzweigungsgrad des Berechnungsbaumes.

Die Laufzeit von M' auf w ist proportional zur Anzahl der Knoten im Berechnungsbaum bis zur Tiefe $p(|w|)$. Diese Anzahl ist beschränkt durch

$$d^{p(|w|)+1} \;=\; 2^{(p(|w|)+1)\cdot \log_2 d} \;=\; 2^{O(p(|w|))}.$$

□

Die große offene Frage der Informatik

$$P = NP?$$

Diese Frage ist bedeutend, weil viele wichtige Probleme, die in der Praxis vorkommen, in NP enthalten sind, wir aber nicht wissen, ob es für diese Probleme effiziente Algorithmen (also Polynomialzeitalgorithmen) gibt.

Einige dieser Problem lernen wir in der nächsten Vorlesung kennen.