
20. Die polynomiell beschränkten Komplexitätsklassen

Nach der Church-Turing-These ist ein Problem genau dann entscheidbar, wenn es Turing-entscheidbar ist, d.h. die charakteristische Funktion des Problems von einer Turingmaschine berechnet werden kann. Wie die Hierarchiesätze zeigen, können die erforderlichen Ressourcen hierbei jedoch so stark anwachsen, dass die Rechnungen praktisch nicht mehr ausführbar sind. Man schränkt daher die Klasse der entscheidbaren Mengen häufig auf die Klasse der *praktisch* oder *tatsächlich* entscheidbaren Mengen ein. Hierbei hat sich die (sicherlich idealisierende) These durchgesetzt, dass diese Klasse mit der Klasse P der in Polynomialzeit von (deterministischen) Turingmaschinen lösbaren Problemen zusammenfällt (*Cooksche These* oder *These von Edmonds*). Da die wechselseitige Simulation der üblichen (deterministischen) Rechnermodelle in polynomialer Zeit möglich ist, ist P maschinenunabhängig, d.h. die in Polynomialzeit lösbaren Probleme sind für alle üblichen (deterministischen!) Rechnermodelle dieselben.

Im Gegensatz zur Berechenbarkeitstheorie, wo man von den meisten interessanten Probleme weiß, ob sie entscheidbar sind, gibt es eine Vielzahl für die Praxis eminent wichtiger Probleme, von denen nicht bekannt ist, ob sie in P liegen, also praktisch lösbar sind. Hatte man es bei den unlösbaren Problemen häufig mit unbeschränkten Suchproblemen zu tun, deren Unentscheidbarkeit man dadurch nachweisen konnte, dass man die Vollständigkeit dieser Probleme für die Klasse der rekursiv aufzählbaren Probleme zeigte, so treffen wir hier nun wiederum auf eine Klasse von Suchproblemen, die nun aber kurze und leicht verifizierbare Lösungen besitzen, wegen der exponentiellen Größe des Suchraums aber eine vollständige Suche in Polynomialzeit nicht erlauben. Von diesen Problemen kann man zeigen, dass sie nichtdeterministisch in Polynomialzeit erkannt werden können, also in NP liegen. Bis heute ist aber offen, ob $P = NP$ oder $P \subsetneq NP$ gilt (*P-NP-Problem*). Man vermutet jedoch, dass P echt in NP enthalten ist, sodass man die Frage, ob ein Problem in P liegt auch unter der Hypothese $P \neq NP$ untersucht. Hier kann man dann durch Einschränkung der effektiven Reduzierbarkeiten auf Polynomialzeit-beschränkte Reduzierbarkeiten einen Vollständigkeitsbegriff für NP so definieren, dass NP-vollständige Probleme nur dann in P liegen, falls – wider Erwarten – $P = NP$ gilt. Durch Nachweis der NP-Vollständigkeit eines Problems kann man also starke Evidenz für die praktische Unlösbarkeit geben. Für die meisten der oben angesprochenen beschränkten Suchprobleme, die in der Praxis aufgetreten sind, konnte man die NP-Vollständigkeit und damit also vermutlich die praktische Unlösbarkeit zeigen.

In diesem Abschnitt werden wir die Klasse NP betrachten, den Begriff der NP-Vollständigkeit einführen und Beispiele NP-vollständiger Mengen vorstellen.

Mit Hilfe des im letzten Abschnitt durchgeführten Vergleichs der Komplexitätsma-

Be können wir die Klassen P und NP wie folgt lokalisieren:

$$\text{LOGSPACE} \subseteq \text{NLOGSPACE} \subseteq \text{P} \subseteq \text{NP} \subseteq \dots \\ \dots \subseteq \text{PSPACE} = \text{NPSPACE} \subseteq \text{EXP} \quad (20.1)$$

Hierbei folgen $\text{PSPACE} = \text{NPSPACE}$ aus dem Satz von Savitch und die nichttrivialen Inklusionen $\text{NLOGSPACE} \subseteq \text{P}$ und $\text{NPSPACE} \subseteq \text{EXP}$ aus Satz 19.3. Auf der anderen Seite folgt aus dem Zeithierarchiesatz, dass P echt in EXP enthalten ist, weshalb zumindest eine der Inklusionen in

$$\text{P} \subseteq \text{NP} \subseteq \text{PSPACE} \subseteq \text{EXP}$$

echt sein muss. Entsprechend folgt die Echtheit zumindest einer der Inklusionen in

$$\text{NLOGSPACE} \subseteq \text{P} \subseteq \text{NP} \subseteq \text{PSPACE}$$

mit dem nichtdeterministischen Platzhierarchiesatz. In der Tat vermutet man die Echtheit aller Inklusionen in (20.1), war aber bislang nicht in der Lage, die Echtheit auch nur einer der Inklusionen zu zeigen. Insbesondere ist die Frage „ $\text{P} = \text{NP}?$ “ noch immer offen, obwohl sich Parallelen im Verhältnis zwischen P und NP sowie zwischen der Klasse REK der rekursiven und der Klasse RA der rekursiv aufzählbaren Mengen zeigen, und $\text{REK} \subsetneq \text{RA}$, wie wir gesehen haben, relativ leicht zu zeigen ist. Die Analogie zwischen diesen Klassen zeigt sich vor allem in der folgenden Charakterisierung der NP-Mengen, die der Charakterisierung der rekursiv aufzählbaren Mengen durch den Projektionssatz entspricht.

20.1 SATZ. *Eine Menge $A \subseteq \Sigma^*$ liegt genau dann in NP, wenn sie die p -beschränkte Projektion einer 2-stelligen Menge $B \subseteq \Sigma^* \times \Sigma^*$ aus P ist. Hierbei ist A die p -beschränkte Projektion der Menge B, wenn es ein Polynom q gibt mit*

$$\forall x \in \Sigma^* (x \in A \Leftrightarrow \exists y (|y| \leq q(|x|) \ \& \ (x, y) \in B)). \quad (20.2)$$

Statt $\exists y (|y| \leq q(|x|) \ \& \ \dots)$ schreibt man auch kurz $\exists^{\leq q(|x|)} y (\dots)$.

BEWEISIDEE. Gilt $A \in \text{NP}$, so gibt es eine $p(n)$ -zeitbeschränkte nichtdeterministische Turingmaschine N , die A erkennt, wobei $p(n)$ ein Polynom ist. Durch geeignete Kodierung der N -Konfigurationen lässt sich für

$$B = \{(x, y) : y \text{ kodiert eine mögliche akzeptierende Rechnung von } N \text{ bei Eingabe } x\} \quad (20.3)$$

zeigen, dass $B \in \text{P}$ gilt. Weiter lässt sich wegen der polynomiellen Zeitschranke für N die Länge der (kodierte) N -Rechnungen ebenfalls polynomiell beschränken, etwa durch das Polynom q (wobei $q \in O(p^2)$). Hiermit ergibt sich für A die gewünschte Projektionsdarstellung (20.2).

Gelte umgekehrt (20.2) für $B \in \text{P}$ und ein Polynom q , so erhält man eine polynomiell zeitbeschränkte nichtdeterministische Turingmaschine zur Erkennung von A wie folgt. Bei Eingabe x erzeugt N zunächst nichtdeterministisch in $\leq q(|x|)$ Schritten jedes Wort y mit $|y| \leq q(|x|)$. (D.h. in jeder möglichen Rechnung wird ein bestimmtes solches y erzeugt. Hierzu reserviert N z.B. eines seiner Bänder und schreibt darauf hintereinander Schritt für Schritt ein nichtdeterministisch gewähltes Zeichen 0,1 oder

B auf das Band. N beendet diese Phase nach dem ersten Schreiben eines Blanks oder falls der parallel mitlaufende Zähler bis $q(|x|)$ (jedes Polynom ist zeitkonstruierbar!) stoppt.) Für das erzeugte y überprüft N dann (durch Simulation einer deterministischen Maschine zur Erkennung von B), ob $(x, y) \in B$ liegt und akzeptiert genau dann, wenn dies der Fall ist. Ist r ein nach Annahme existierendes Polynom mit $B \in \text{DTIME}(r(n))$, so lässt sich die Laufzeit der Maschine N durch das Polynom

$$p(n) = q(n) + r(2n + q(n) + 1)$$

abschätzen. Hierbei gehen wir davon aus, dass die 2-stellige Menge B durch die 1-stellige Menge $B' = \{1^{|x|}0xy : (x, y) \in B\}$ kodiert ist, d.h. einer Eingabe (x, y) für B die Länge $|x| + 1 + |x| + |y| = 2|x| + |y| + 1$ zugeordnet wird. \square

Satz 20.1 hilft häufig beim Nachweis, dass ein Problem in NP liegt. Wir geben hierfür zwei Beispiele.

Wir betrachten zunächst das *Erfüllbarkeitsproblem* für aussagenlogische Formeln in konjunktiver Normalform (kNF), abgekürzt SAT (= Satisfiability). Die Menge CNF der aussagenlogischen Formeln in konjunktiver Normalform ist induktiv definiert. Wir gehen von einem abzählbaren Vorrat x_1, x_2, x_3, \dots von (*Aussagen-*)*Variablen* und den Wahrheitswerten (*Konstanten*) 0 (= falsch) und 1 (= wahr) aus. Ein *Literal* l ist eine Variable ($l = x$), eine negierte Variable ($l = \neg x$), eine Konstante ($l = i$) oder eine negierte Konstante ($l = \neg i$). Eine \vee -*Klausel* c ist die endliche Disjunktion von Literalen ($c = l_1 \vee l_2 \vee \dots \vee l_n, n \geq 1$). Eine *Formel* α in kNF ist die endliche Konjunktion von \vee -Klauseln ($\alpha = c_1 \& c_2 \& \dots \& c_m, m \geq 1$). Mit CNF bezeichnen wir die Menge aller Formeln in kNF. Dabei gehen wir davon aus, dass Formeln als Wörter über dem Alphabet $\Sigma = \{0, 1, \neg, \vee, \&\}$ dargestellt sind, wobei wir die Variable x_n durch das Binärwort $0^{\text{bin}(n)}$ repräsentieren. (Mit der Länge $|\alpha|$ einer Formel α meinen wir dann die Länge des zugehörigen Wortes.) Eine Formel, die keine Variablen enthält, heißt *geschlossen*. Enthält die Formel keine Konstanten, so heißt sie *pur*. Einer geschlossenen Formel α kann man einen Wahrheitswert $w(\alpha)$ zuordnen, indem man $w(i) = i$ für die Konstante $i = 0, 1$ setzt, und w dann durch

$$\begin{aligned} w(\neg i) &= 1 - i \\ w(l_1 \vee \dots \vee l_n) &= \max(w(l_1), \dots, w(l_n)) \\ w(c_1 \& \dots \& c_m) &= \min(w(c_1), \dots, w(c_m)), \end{aligned}$$

d.h. durch die übliche Interpretation der Junktoren \neg, \vee und $\&$ auf Literale, Klauseln und kNF-Formeln fortsetzt. Der Wahrheitswert einer geschlossenen kNF-Formel lässt sich durch einmaliges Lesen der Formel von links nach rechts berechnen (wobei sich gleichzeitig feststellen lässt, ob überhaupt eine (geschlossene) Formel vorliegt, d.h.

$$W = \{\alpha \in \Sigma^* : \alpha \in \text{CNF} \& \alpha \text{ geschlossen} \& w(\alpha) = 1\} \in \text{DTIME}(O(n)).$$

Eine *Bewertung* oder *Belegung* B einer beliebigen Formel α ist eine Abbildung von der Menge $\text{Var}(\alpha)$ der in α vorkommenden Variablen in $\{0, 1\}$, ordnet also jeder Variablen in α einen Wahrheitswert zu. Die Bewertung B *macht* α *wahr*, wenn $w(\alpha_B) = 1$ gilt, wobei $\alpha_B = \alpha_{x_1, \dots, x_n}[B(x_1), \dots, B(x_n)]$ die geschlossene Formel ist, die man aus α erhält, wenn man simultan die vorkommenden Variablen x durch $B(x)$ ersetzt. Die Formel α ist *erfüllbar*, wenn es eine Belegung von α gibt, die α wahr macht, und wir definieren

$$\text{SAT} = \{\alpha \in \text{CNF} : \alpha \text{ erfüllbar}\}.$$

20.2 LEMMA. $\text{SAT} \in \text{NP}$.

BEWEIS. Für $\alpha \in \text{CNF}$, $|\alpha| = n$, können wir $\text{Var}(\alpha) = \{x_1, \dots, x_m\}$ in Quadratzeit (oder schneller) bestimmen, und offensichtlich gilt $|\text{Var}(\alpha)| = m \leq n$. Jede mögliche Belegung B von α entspricht dann einem Binärwort $w = B(x_1) \dots B(x_m)$ der Länge $m \leq n$. Es gilt also

$$\begin{aligned} \alpha \in \text{SAT} &\Leftrightarrow \exists^{\leq |\alpha|} B (B \text{ macht } \alpha \text{ wahr}) \\ &\Leftrightarrow \exists^{\leq |\alpha|} B ((\alpha, B) \in \tilde{W}), \end{aligned}$$

wobei

$$\tilde{W} = \{(\alpha, B) : \alpha_B \in W\}$$

für die oben definierte Menge W . Da aus $W \in \text{P}$ leicht $\tilde{W} \in \text{P}$ folgt, zeigt dies $\text{SAT} \in \text{NP}$ nach Satz 20.1. \square

Als zweites Beispiel betrachten wir das *Knotenüberdeckungsproblem* VC (vertex cover) in der Graphentheorie. Ein endlicher (ungerichteter) *Graph* $G = (V, E)$ besteht aus einer endlichen Menge V von Knoten und einer Menge E von Kanten, die einen Teil der Knoten verbinden. D.h. $E \subseteq V \times V$ ist eine 2-stellige, symmetrische Relation. Als Knotenmenge nehmen wir hierbei stets ein Anfangsstück $\{0, \dots, n-1\}$ der natürlichen Zahlen. Die Kanten repräsentieren wir dann durch die sog. *Adjazenzmatrix* über $\{0, 1\}$, wobei

$$M[i, j] = 1 \Leftrightarrow (i, j) \in E$$

gilt. Durch Linearisieren der Matrix kann man dann endlichen Graphen Binärwörtern zuordnen, deren Länge quadratisch in der Anzahl der Knoten ist.

Eine *Knotenüberdeckung* U des Graphen $G = (V, E)$ ist eine Teilmenge U der Knotenmenge V , sodass von jeder Kante von G zumindest einer der Endpunkte zu U gehört, d.h.

$$\forall i, j \in V ((i, j) \in E \Rightarrow i \in U \vee j \in U)$$

gilt. Das *Knotenüberdeckungsproblem* fragt nach der Existenz einer Knotenüberdeckung U eines gegebenen Graphen $G = (V, E)$, die eine vorgegebene Größe k hat. D.h.

$$\text{VC} = \{(G, k) : G \text{ besitzt eine Knotenüberdeckung } U \text{ mit } \|U\| = k\}.$$

Da hierbei $G = (V, E)$ mit $\|V\| = n$ wie oben bemerkt durch ein Wort der Länge n^2 dargestellt wird und o.B.d.A. $k \leq n$ angenommen werden kann, spielt die Wahl der Repräsentation von k keine Rolle.

20.3 LEMMA. $\text{VC} \in \text{NP}$.

BEWEIS. Seien $G = (V, E)$ und k mit $k \leq n = \|V\|$ gegeben, und sei M die Adjazenzmatrix von G . Eine Menge $U \subseteq V = \{0, \dots, n-1\}$ ist genau dann eine Knotenüberdeckung von G , wenn

$$\forall i, j < n (M[i, j] = 1 \Rightarrow i \in U \text{ oder } j \in U),$$

was sich in $O(n^2)$ Schritten testen lässt. Es folgt, dass die Menge

$$B = \{(G, U, k) : U \text{ Knotenüberdeckung der Größe } k \text{ des Graphen } G\}$$

in $\text{DTIME}(O(n))$ liegt. Da

$$(G, k) \in \text{VC} \Leftrightarrow \exists \leq^k U ((G, U, k) \in B),$$

folgt hieraus $\text{VC} \in \text{NP}$ mit Satz 20.1. \square

Um zu zeigen, dass – unter der Voraussetzung, dass $P \neq \text{NP}$ gilt – diese (und andere) NP-Probleme nicht in P liegen, hat man folgende Polynomialzeit-Variante der many-one-Reduzierbarkeit eingeführt.

20.4 DEFINITION. Eine Menge $A \subseteq \Sigma^*$ ist auf eine Menge $B \subseteq \Sigma^*$ *p-m-reduzierbar* ($A \leq_m^P B$), wenn es eine Funktion $f : \Sigma^* \rightarrow \Sigma^*$ in FP (d.h. der Klasse der in Polynomialzeit von einer deterministischen Turingmaschine berechenbaren Funktionen) gibt, sodass

$$\forall x \in \Sigma^* (x \in A \Leftrightarrow f(x) \in B)$$

gilt, d.h. $c_A = c_B \circ f$.

Die Definition von \leq_m^P unterscheidet sich von der klassischen Definition von \leq_m nur dadurch, dass rekursiv (d.h. berechenbar) durch Polynomialzeit-Turingberechenbar (d.h. tatsächlich berechenbar) ersetzt wird. Statt der Nichtrekursivität (Unentscheidbarkeit) kann man entsprechend nun die Nichtmitgliedschaft in P (praktische Nichtentscheidbarkeit) mit Hilfe von \leq_m^P zeigen, indem man die über \leq_m gemachten Beobachtungen entsprechend auf \leq_m^P überträgt und korrespondierend Härte- und Vollständigkeitskonzepte einführt.

20.5 LEMMA. (a) \leq_m^P ist eine Präordnung, d.h. reflexiv und transitiv.

(b) Gilt $A \leq_m^P B$ und $B \in P$, so gilt auch $A \in P$.

(c) Gilt $A \leq_m^P B$ und $A \notin P$, so gilt auch $B \notin P$.

BEWEIS. Man zeigt dies wie die entsprechenden Aussagen für \leq_m , wobei man beachtet, dass die Klasse der Polynome gegen explizite Definitionen abgeschlossen ist. Wir begnügen uns daher mit dem Beweis von (b). Gelte $A \leq_m^P B$ via f und sei $B \in P$. Dann gibt es Polynome p und q , sodass $f \in \text{FDTIME}(p(n))$ und $B \in \text{DTIME}(q(n))$ gilt. Da nach Wahl von f für jedes Wort x ($|x| = n$)

$$c_A(x) = c_B(f(x))$$

gilt, kann man $c_A(x)$ berechnen, indem man zunächst $f(x)$ berechnet ($p(n)$ Schritte), und dann $c_B(f(x))$ ($q(|f(x)|)$ Schritte). Wegen der Zeitschranke $p(n)$ für f gilt jedoch $|f(x)| \leq p(|x|)$, weshalb die Berechnung von $c_A(x)$ in $O(p(n) + q(p(n)))$, d.h. polynomial vielen Schritten möglich ist. \square

20.6 DEFINITION. Eine Menge A ist *p-m-hart* für eine Komplexitätsklasse C , falls $B \leq_m^P A$ für alle $B \in C$ gilt. Gilt zusätzlich noch, dass A selbst in C liegt, so ist A *p-m-vollständig* für C .

Statt p - m -hart bzw. p - m -vollständig für $C \subseteq \text{REK}$ sagen wir einfacher C -hart und C -vollständig. Aus Lemma 20.5(c) folgt direkt, dass für jede Komplexitätsklasse C , die nicht in P enthalten ist, die C -harten – und daher auch die C -vollständigen – Mengen nicht in P liegen. (Die Umkehrung gilt trivialerweise.) Darüberhinaus besitzt jede abzählbare Klasse $C = \{C_n : n \geq 1\}$ (also insbesondere jede Komplexitätsklasse $C \subseteq \text{REK}$) C -harte Probleme, wie z.B. das Problem $H_C = \{1^n 0x : x \in C_n\}$, da $C_n \leq_m^P H_C$ via $f_n(x) = 1^n 0x$ gilt¹. Insbesondere gilt also:

20.7 LEMMA. Falls $P \neq \text{NP}$ und A NP-vollständig ist, so gilt $A \notin P$

Um zu zeigen, dass es überhaupt NP-vollständige Mengen gibt, betrachten wir das *eingeschränkte Halteproblem* für nichtdeterministische 1-Band-Turingmaschinen:

$$K_{[1,nd]}^b := \{1^e 0x 01^n : U_{1,nd} \text{ akzeptiert } (e, x) \text{ in } \leq n \text{ Schritten}\}.$$

Hierbei ist $U_{1,nd}$ die universelle Maschine für die nichtdeterministischen (normierten) 1-Band-Turingmaschinen.

20.8 SATZ. $K_{[1,nd]}^b$ ist NP-vollständig.

BEWEIS. Dass $K_{[1,nd]}^b$ in NP liegt, kann man mit Satz 20.1 zeigen. Es genügt hierzu zu beobachten, dass $K_{[1,nd]}^b$ die beschränkte Projektion der P -Menge

$$B = \{(1^e 0x 01^n, y) : y \text{ kodiert eine mögliche akzeptierende Rechnung von } U_{1,nd} \text{ bei Eingabe } (e, x), \text{ deren Länge } \leq n \text{ ist}\}$$

ist, nämlich

$$w \in K_{[1,nd]}^b \Leftrightarrow \exists \leq |w|^2 y ((w, y) \in B).$$

Zum Nachweis der NP-Härte von $K_{[1,nd]}^b$ müssen wir für gegebenes $A \in \text{NP}$ nachweisen, dass $A \leq_m^P K_{[1,nd]}^b$ gilt. Wegen $A \in \text{NP}$ gibt es eine nichtdeterministische Turingmaschine N , die A erkennt, und die für ein geeignetes Polynom $p(n)$ -zeitbeschränkt ist. Da die Reduktion auf ein Band, die Normierung und schließlich die Simulation durch die universelle Maschine $U_{1,nd}$ nur zu einem quadratischen Zeitverlust führen, gibt es eine Zahl e und ein Polynom $q(n) \in O(p(n^2))$ mit

$$\begin{aligned} x \in A &\Leftrightarrow N \text{ akzeptiert } x \\ &\Leftrightarrow N \text{ akzeptiert } x \text{ in } \leq p(|x|) \text{ Schritten} \\ &\Leftrightarrow U_{1,nd} \text{ akzeptiert } (e, x) \\ &\Leftrightarrow U_{1,nd} \text{ akzeptiert } (e, x) \text{ in } \leq q(|x|) \text{ Schritten} \end{aligned}$$

Nach Definition von $K_{[1,nd]}^b$ heißt dies aber gerade, dass

$$x \in A \Leftrightarrow 1^e 0x 01^{q(|x|)} \in K_{[1,nd]}^b$$

gilt, woraus $A \leq_m^P K_{[1,nd]}^b$ via $f(x) = 1^e 0x 01^{q(|x|)}$ folgt. \square

Weitere Beispiele für NP-vollständige Probleme sind die oben eingeführten Probleme SAT und VC. Wir zeigen dies hier nur für SAT.

¹Nicht jede Klasse $C \subseteq \text{REK}$ besitzt jedoch vollständige Probleme. Z.B. besitzt REK selbst keine vollständigen Probleme. Hierzu beobachtet man, dass es zu jeder rekursiven Menge A eine rekursive Menge B mit $B \not\leq_m^P A$ gibt. Ist nämlich A in $\text{DTIME}(t(n))$ (o.B.d.A. t monoton), so gilt, dass $\{C : C \leq_m^P A\} \subseteq \text{DTIME}(t(\text{poly}(n))) \subseteq \text{DTIME}(t(2^n))$. Nach dem Zeithierarchiesatz gilt jedoch $\text{DTIME}(t(2^n)) \not\subseteq \text{REK}$.

20.9 SATZ. (SATZ VON COOK) SAT ist NP-vollständig.

BEWEIS. Wegen Lemma 20.2 genügt es zu zeigen, das SAT NP-hart ist. Sei also $A \in \text{NP}$ gegeben. Um $A \leq_m^P \text{SAT}$ zu zeigen, beschreiben wir die Rechnungen einer nichtdeterministischen, polyzeitbeschränkten Turingmaschine N , die A erkennt, durch CNF-Formeln. D.h. wir ordnen jeder Eingabe x eine Formel $\alpha_x \in \text{CNF}$ so zu, dass α_x genau dann erfüllbar ist, wenn N bei Eingabe x eine akzeptierende Rechnung besitzt. Da sich α_x in Polynomialzeit aus x berechnen lassen wird, wird hieraus $A \leq_m^P \text{SAT}$ vermöge der Reduktionsfunktion $f(x) = \alpha_x$ folgen.

Zur Vereinfachung der Definition der Formeln α_x gehen wir dabei von gewissen Normeigenschaften der nichtdeterministischen Maschine N aus, die A akzeptiert: Wegen des Bandreduktionssatzes können wir annehmen, dass N eine Einbandmaschine ist. Sind Z, Γ, P Zustandsmenge, Bandalphabet und Programm von N , so können wir $Z = \{0, 1, \dots, m\}$ voraussetzen, wobei 0 der Startzustand und 1 der einzige akzeptierende Zustand von N ist. Den Nichtdeterminismus von N können wir auf den Nachfolgezustand beschränken, d.h. für Instruktionen aus P

$$(z, a, a', B', z') \neq (z, a, a'', B'', z'') \Rightarrow a' = a'' \ \& \ B' = B'' \quad (20.4)$$

fordern. (Wird dies durch Instruktionen wie oben verletzt, ersetzen wir diese durch

$$(z, a, a, S, z_1), (z_1, a, a', B', z'), (z, a, a, S, z_2), (z_2, a, a'', B'', z''),$$

wobei z_1, z_2 zwei neue Zustände sind.)

Für die Definition der Formeln α_x erweitern wir das Programm P zu einem Programm P' , indem wir die Stoppzustände s von P in absorbierende Zustände verwandeln, indem wir Instruktionen (s, a, a, S, s) hinzufügen. Statt zu stoppen gerät N also in eine Endlosschleife, ohne jedoch die Stoppkonfiguration zu verändern. Jede Rechnung der derart modifizierten Maschine N' ist also unendlich, und für eine polynomielle Zeitschranke $p(n)$ für N gilt

$$\begin{aligned} x \in A &\Leftrightarrow \text{Es gibt eine akzeptierende Rechnung von } N \\ &\quad \text{bei Eingabe } x \text{ (der Länge } \leq p(|x|)). \\ &\Leftrightarrow \text{Es gibt eine Rechnung von } N' \text{ bei Eingabe } x, \\ &\quad \text{die nach } p(|x|) \text{ Schritten den Zustand 1 erreicht hat.} \end{aligned} \quad (20.5)$$

Wir kommen nun zur Definition von α_x für gegebenes $x = x_1 \dots x_n$, $|x| = n$. In α_x kommen die Variablen

$$z_{t,i}, p_{t,j}, b_{t,j,a} \quad (0 \leq t \leq p(n), i \in Z, -p(n) \leq j \leq p(n), a \in \Gamma)$$

vor, die bezüglich einer beliebigen aber festen möglichen Rechnung von N bei Eingabe x interpretiert werden als die Aussagen

$$\begin{aligned} z_{t,i} &= \text{Der Zustand zum Zeitpunkt } t \text{ ist } i. \\ p_{t,j} &= \text{Die Position des Arbeitsfeldes zum Zeitpunkt } t \text{ ist } j. \\ b_{t,j,a} &= \text{Die Beschriftung von Feld } j \text{ zum Zeitpunkt } t \text{ ist } a. \end{aligned}$$

Jede mögliche Rechnung definiert auf diese Art eine Belegung der Variablen in α_x , und die Definition von α_x wird sicherstellen, dass diese Belegung α_x genau dann wahr macht, wenn die Rechnung akzeptiert. Umgekehrt wird jede Belegung der Variablen, die α wahr macht, über die obige Interpretation eine akzeptierende Rechnung definieren.

Die Formel α_x ist Konjunktion folgender Klauseln, die wir nach ihrer intendierten Bedeutung in Gruppen zusammenfassen (wobei stets $t \in [0, p(n)]$, $i \in Z$, $j \in [-p(n), p(n)]$ und $a \in \Gamma$):

1. Beschreibung der Startkonfiguration

$$\begin{array}{ll} z_{0,0} & \\ p_{0,0} & \\ b_{0,m,x_m} & (1 \leq m \leq n) \\ b_{0,l,B} & (l \in [-p(n), p(n)] - [1, n]) \end{array}$$

2. Beschreibung der Nachfolgekonfiguration²

a) Zustand

$$(z_{t,i} \& p_{t,j} \& b_{t,j,a}) \rightarrow \bigvee_{i' \in Z_{(a,i)}} z_{t+1,i'}$$

Hierbei ist $Z_{(a,i)}$ die Menge der möglichen Nachfolgezustände von i bei Inschrift a des Arbeitsfeldes, d.h.

$$Z_{(a,i)} = \{i' : \exists a' \in \Gamma \exists B \in \{L, R, S\} ((i, a, a', B, i') \in P')\}.$$

b) Position des Arbeitsfeldes

$$(z_{t,i} \& p_{t,j} \& b_{t,j,a}) \rightarrow p_{t+1,j+\hat{B}}$$

Hierbei ist B die nach (20.4) eindeutig festgelegte Bewegung, die N im Zustand i bei Inschrift a des Arbeitsfeldes ausführt, und \hat{B} durch $\hat{L} = -1$, $\hat{S} = 0$ und $\hat{R} = 1$ festgelegt liegt.

c) Inschrift des Bandes

$$\begin{array}{ll} (z_{t,i} \& p_{t,j} \& b_{t,j,a}) & \rightarrow b_{t+1,j,a'} \\ (z_{t,i} \& p_{t,j} \& b_{t,j',a''}) & \rightarrow b_{t+1,j',a''} \end{array}$$

Hierbei ist a' die nach (20.4) eindeutig festgelegte Neubeschriftung des (alten) Arbeitsfeldes von N' , wenn N' im Zustand i und a die alte Beschriftung ist; $j' \in [-p(n), p(n)] - \{j\}$; und $a'' \in \Gamma$.

3. Akzeptanz

$$z_{p(n),1} \quad (\text{vgl. (20.5)})$$

4. Eindeutigkeitsregeln

$$\begin{array}{lll} z_{t,i} & \rightarrow & \neg z_{t,i'} \quad (i' \in Z - \{i\}) \\ p_{t,j} & \rightarrow & \neg p_{t,j'} \quad (j' \in [-p(n), p(n)] - \{j\}) \\ b_{t,j,a} & \rightarrow & \neg b_{t,j,a'} \quad (a' \in \Gamma - \{a\}) \end{array}$$

²Zur Erhöhung der Übersichtlichkeit geben wir zu Klauseln äquivalente Formeln an, wobei wir die Implikation $\beta \rightarrow \gamma$ als Abkürzung für $\neg\beta \vee \gamma$ verwenden. Die Äquivalenz zu einer Klausel folgt mit der DeMorganschen Regel $\neg(\beta_1 \& \dots \& \beta_n) = \neg\beta_1 \vee \dots \vee \neg\beta_n$.

Wie man leicht sieht (wenn der formale Nachweis auch sehr mühselig ist), lässt sich α_x in Polynomialzeit (Quadratzeit) berechnen. Den formalen Nachweis, dass

$$x \in A \Leftrightarrow \alpha_x \in \text{SAT}$$

gilt, führt man wie folgt. Zum Nachweis der Richtung „ \Rightarrow “ genügt es wegen (20.5) zu zeigen, dass α_x erfüllbar ist, falls es eine mögliche Rechnung $R = C_0, C_1, \dots, C_{p(n)}$ von N' bei Eingabe x gibt, die im Zustand 1 endet. Definiert man zu dieser Rechnung eine Belegung B der Variablen in α_x gemäß der oben genannten intendierten Bedeutung derselben, d.h.

$$\begin{aligned} B(z_{t,i}) = 1 &\Leftrightarrow R \text{ ist zum Zeitpunkt } t \text{ im Zustand } i. \\ B(p_{t,j}) = 1 &\Leftrightarrow \text{Die Position des Arbeitsfeldes zum Zeitpunkt } t \\ &\text{der Rechnung } R \text{ ist } j. \\ B(b_{t,j,a}) = 1 &\Leftrightarrow \text{Das Feld } j \text{ ist zum Zeitpunkt } t \text{ der Rechnung } R \\ &\text{mit } a \text{ beschriftet,} \end{aligned} \quad (20.6)$$

so macht diese Belegung die Formel α_x wahr. Für die Eindeigkeitsklauseln ist dies trivial. Für die anderen Klauseln zeigt man dies durch (eine ebenfalls evidente) Induktion nach dem vorkommenden Index t ($0 \leq t \leq p(n)$).

Zum Nachweis der Umkehrung „ \Leftarrow “ sei eine Belegung B gegeben, die α_x wahr macht. Durch Induktion nach t zeigt man, dass dann durch (20.6) die Belegung B eine Rechnung von N' bei Eingabe x beschreibt, die wegen der Akzeptanzklausel im Zustand 1 endet. (Für diesen Teil des Beweises ist die Annahme (20.4) wichtig. Da z.B. Nachfolgezustand und Nachfolgeposition des Arbeitsfeldes durch unabhängige Klauseln definiert sind, könnte man B sonst so definieren, dass diese Parameter für verschiedene Rechnungen gewählt werden, also nicht zusammenpassen.) \square

Analog zu dem entsprechenden Ergebnis in der Berechenbarkeitstheorie gilt

$$A \text{ NP-hart} \ \& \ A \leq_m^P B \Rightarrow B \text{ NP-hart.}$$

Den Hårteteil in einem Vollständigkeitsbeweis für ein Problem B führt man daher meist, indem man ein bekanntes, NP-vollständiges Problem A auf B reduziert. So geht man auch beim Nachweis der NP-Vollständigkeit für VC vor, indem man SAT auf VC reduziert. Eine gute Zusammenstellung der wichtigsten NP-vollständigen Probleme sowie typische, instruktive Beispiele für Reduktionsbeweise (wie der von SAT auf VC) finden sich in dem Buch von Garey und Johnson. Aus Zeitgründen können wir hier auf dieses wichtige Thema nicht weiter eingehen. Zum Abschluss soll jedoch noch bemerkt werden, dass in der Praxis meist nicht die hier behandelten NP-vollständigen Entscheidungsprobleme, sondern die zugehörigen Such- oder (bei parametrisierten Problemen wie VC) Optimierungsprobleme interessieren. So möchte man im Falle von SAT für eine erfüllbare Formel eine sie wahr machende Belegung angeben, und im Falle von VC möchte man eine Knotenüberdeckung minimaler Größe finden.

Man kann jedoch zeigen, dass diese Probleme im Wesentlichen äquivalent sind. Die bekannten NP-vollständigen Probleme besitzen nämlich eine gewisse Selbstreduktionseigenschaft, die es erlaubt, Entscheidungsverfahren effizient in korrespondierende Such- bzw. Optimierungsverfahren umzuwandeln. (Mehr hierzu in den Übungen.)