Berechenbarkeit und Komplexitätstheorie

Wintersemester 2021/2022
Aufgabenblatt 8
Abgabe: 21. Januar 2022 um 12 Uhr

Definition(en)

Analog zum Begriff der NP-Schwere nennen wir eine Sprache L P-schwer (oder P-hart), wenn alle Sprachen aus P auf sie polynomial reduzierbar sind. Eine Sprache L heißt P-vollständig, wenn sie in P liegt und P-schwer ist.

Aufgabe 8.1 *(4 Punkte)*

Zeigen Sie: $\{w \cdot ↡ \cdot w \colon w \in \{ʔ, ᐳ\}^*\}$ ist in P.

Lösung

Hierbei müssen wir eine TM konstruieren, die (deterministisch) in Polynomzeit entscheidet, ob ein Wort von der Form $w \cdot ↡ \cdot w$ ist, also ob das Wort aus zwei gleichen Wortteilen besteht, die durch das Trennsymbol ↡ getrennt sind. Folgende (grobe) Arbeitsweise beschreibt eine solche TM:

- Markiere den linken und den rechten Rand mit einem eigenen Symbol, bspw. Ħ.
- Lies das linkeste (nicht-Rand-)Symbol ein und speichere dieses (Zustand oder zweites Arbeitsband).
- Überschreibe das Feld mit einem □ und gehe danach zum ersten Symbol (ungleich □) nach dem Trennsymbol.
- Überprüfe, ob dieses Symbol gleich dem gespeicherten ist. Wenn nein: Lehne ab. Wenn ja: Überschreibe das Feld mit einem □ und gehe wieder zum linkesten Symbol (ungleich □ bzw. Ħ).
- Wiederhole den Vorgang, bis das linkteste Symbol das Trennzeichen ↡ ist. Akzeptiere, falls das Band rechts des Trennsymbols auch leer ist, sonst lehne ab.

Der erste Schritt (markieren der Ränder) hilft dabei, zu testen ob das Band rechts des Trennsymbols ↡ leer ist (d. h. bei der beschriebenen Arbeitsweise würde auch nur der rechte Rand reichen). Ohne diese Markierung würde die TM sonst nicht wissen, wie weit sie nach rechts laufen muss, um ein potentielles Zeichen aus $\{ʔ, ᐳ\}$ zu finden. Statt den Randsymbolen kann man sich auch andere Möglichkeiten überlegen, wie man erkennt, wo das Wort ursprünglich zu Ende war (spezielle Blanksymbole, mitzählen der Wortlänge, beim Überprüfen des rechten Symbols testen, ob es das letzte Zeichen des Wortes ist, …).

Um die Ränder des Wortes zu markieren müssen wir zweimal über die gesamte Eingabe laufen (einmal für den rechten Rand und einmal, um den Kopf wieder auf das erste Symbol in w zu bewegen). Bei den restlichen Schritten (der "Hauptschleife") laufen wir immer über die "halbe" Eingabe (Länge $|w|$) um zwei Zeichen zu vergleichen. Es werden insgesamt $|w|$ viele Vergleiche durchgeführt. Um zu testen, ob der rechte Teil der Eingabe leer ist, muss die TM potentiell nochmal über die halbe Eingabe laufen. Für ein Wort der Länge n ergibt sich dann eine Laufzeit von $2 \cdot n + |w|^2 + |w| = \left(\frac{1}{2} \cdot n\right)^2 + \frac{5}{2} \cdot n \in \mathcal{O}(n^2)$ gilt, haben wir eine in Polynomzeit arbeitende TM konstruiert.

Aufgabe 8.2 *(4 Punkte)*

Zeigen Sie, dass jede Sprache $A \in$ P (außer $\emptyset$ und Σ^*) P-vollständig ist.

Lösung

Nach Voraussetzung ist $A \in \mathsf{P}$, also bleibt nur die P-Schwere zu zeigen. Sei $B \in \mathsf{P}$ beliebig. Seien $a \in A$ und $b \notin A$ beliebig (da $\emptyset \subsetneq A \subsetneq \Sigma^*$ existieren beide Elemente). Wir betrachten folgende Reduktionsfunktion f berechnet durch TM M_f:

- Simuliere M_B mit der Eingabe w
- Falls M_B akzeptiert, schreibe a auf das Band
- Falls M_B ablehnt, schreibe b auf das Band

Damit gilt $w \in B \equiv f(w) \in A$. Die beiden letzten Schritte sind jeweils in konstanter Zeit möglich (M_B berechnet eine charakteristische Funktion und die Ausgabe ist entweder 0 oder 1 – also nur ein Zeichen, dass überschrieben werden muss). Der erste Schritt läuft in Polynomzeit, da nach Voraussetzung $B \in \mathsf{P}$, also M_B eine Polynomzeit-DTM ist. Da B beliebig gewählt war heißt das, dass A P-schwer und damit auch P-vollständig ist.

(*Bemerkung:* Diese Aufgabe zeigt, dass die obige Definition zur P-Schwere überflüssig ist, da die Reduktionsfunktion selbst genug Ressourcen hat, um das gegebene Problem zu lösen. Typischerweise definiert man P-Schwere darum bzgl. einer "logarithmischen" statt einer polynomialen Reduktion.)

Aufgabe 8.3 *(4 + 3 Punkte)*

a) Zeigen Sie, dass SAT in NP ist.

(*Hinweis:* Nutzen Sie aus, dass Formeln entweder innerhalb eines Klammerpaares stehen oder auf ein Negationszeichen folgen und Sie somit immer eine "innerste Formel" finden können, die Sie direkt auswerten können.)

b) Spielen Sie Ihren Algorithmus mit der Beispielformel $\phi = ((v_1 \vee v_2) \wedge \neg v_2)$ durch.

Lösung

a) An dieser Stelle reicht ein einfacher "guess and check"-Algorithmus. Wir raten uns zu einer gegebenen Formel eine Belegung der Variablen und überprüfen dann in Polynomzeit, ob die Formel unter der geratenen Belegung WAHR ist. Etwas Formaler: Sei ϕ eine aussagenlogische Formel mit den Variablen $v_1, ..., v_n$. Unsere NTM M arbeitet folgendermaßen:

- Für $1 \leq i \leq n$: Rate eine Belegung für v_i.
- Ersetze in ϕ jedes Vorkommen von v_i durch die zugehörige, geratene Belegung.
- Suche eine "innerste Formel" und ersetze den Ausdruck durch den passenden Wahrheitswert wie folgt:
  - $\neg 0$ wird ersetzt durch 1
  - $\neg 1$ wird ersetzt durch 0
  - $(0 \vee 0)$ wird ersetzt durch 0
  - $(0 \vee 1)$ wird ersetzt durch 1
  - $(1 \vee 0)$ wird ersetzt durch 1
  - $(1 \vee 1)$ wird ersetzt durch 1
  - $(0 \wedge 0)$ wird ersetzt durch 0

  - $(0 \wedge 1)$ wird ersetzt durch 0
  - $(1 \wedge 0)$ wird ersetzt durch 0
  - $(1 \wedge 1)$ wird ersetzt durch 1

  In diesem Schritt müsste man formal noch ergänzen, dass die entstehenden $\square$-Symbole zwischen den Formel-Symbolen "gelöscht" werden, oder dass der Ersetzungsschritt diese ignoriert. (Also bspw. Statt $(1 \vee 0)$ durch 1, $(\square^*1\square^* \vee \square^*0\square^*)$ durch $1\square^*$ ersetzen.)
- Wiederhole den vorigen Schritt solange, bis keine inneren Formeln mehr vorhanden sind.
- Akzeptiere, wenn zum Schluss eine 1 auf dem Band steht.

Das Raten der Belegungen und ersetzen der Werte erfolgt in konstanter Zeit (oder Polynomzeit, je nach "Interpretation" des Nichtdeterminismus). Sowohl das Ersetzen einer innersten Formel als auch das Finden einer solchen Formel ist in Polynomzeit möglich (hier muss die TM einmal über das Arbeitsband laufen). Da durch das Ersetzen der innersten Formeln die gesamte Formel immer kürzer wird, endet der Algorithmus auch irgendwann.

b) Wenn man sich streng an die Syntax aus dem Skript hält, werden die Variablen binär kodiert (und brauchen somit mehr als ein Symbol). Wenn wir annehmen, dass entstehende leere Felder nicht gelöscht werden, ergibt sich folgender Ablauf:

- | (| (| v | 0 | 1 | $\vee$ | v | 1 | 0 |) | $\wedge$ | $\neg$ | v | 1 | 0 |) |
- Geratene Belegung: $v_1 = 1$, $v_2 = 0$
- | (| (| 1 | $\square$ | $\square$ | $\vee$ | 0 | $\square$ | $\square$ |) | $\wedge$ | $\neg$ | 0 | $\square$ | $\square$ |) |
- | (| (| 1 | $\square$ | $\square$ | $\vee$ | 0 | $\square$ | $\square$ |) | $\wedge$ | 1 | $\square$ | $\square$ | $\square$ |) |
- | (| 1 | $\square$ | $\square$ | $\square$ | $\square$ | $\square$ | $\square$ | $\square$ | $\square$ | $\wedge$ | 1 | $\square$ | $\square$ | $\square$ |) |
- | 1 | $\square$ | $\square$ | $\square$ | $\square$ | $\square$ | $\square$ | $\square$ | $\square$ | $\square$ | $\square$ | $\square$ | $\square$ | $\square$ | $\square$ | $\square$ |

Wenn man Variablen als einzelne Symbole kodiert und leere Zellen "löscht", ergibt sich folgender Ablauf:

- | (| (| v_1 | $\vee$ | v_2 |) | $\wedge$ | $\neg$ | v_2 |) |
- Geratene Belegung: $v_1 = 1$, $v_2 = 0$
- | (| (| 1 | $\vee$ | 0 |) | $\wedge$ | $\neg$ | 0 |) |
- | (| 1 | $\wedge$ | $\neg$ | 0 |) |
- | (| 1 | $\wedge$ | 1 |) |
- | 1 |