Institut für Informatik
Prof. Dr. J. Rothe

Wintersemester 2023/24
Übungsblatt 10

Lösungsvorschläge

Kryptokomplexität 1

Bearbeitungszeit: 12. Dezember bis 20-22. Dezember
Verantwortlich: Roman Zorn

Aufgabe 1 : Erfüllbarkeit von booleschen Funktionen

Sei $f\colon \{0,1\}^n \to \{0,1\}$ eine boolesche Funktion und betrachten Sie die folgenden Repräsentationen für boolesche Funktionen:

(i) *Boolesche Formel*: Für jedes $f\colon \{0,1\}^n \to \{0,1\}$ existiert eine boolesche Formel φ mit n Variablen, so dass $f(x_1,\ldots,x_n) = 1$ genau dann gilt, wenn φ ausgewertet mit der Belegung $(x_1,\ldots,x_n)$ wahr ist.

(ii) *Wahrheitstabelle*: In der Wahrheitstabelle für f ist zeilenweise für jede möglich Eingabe $(x_1,\ldots,x_n) \in \{0,1\}^n$ von f der Funktionswert $f(x_1,\ldots,x_n)$ abgelegt.

Wir nehmen an, dass konkrete Kodierungen für beide Repräsentationen existieren. Bearbeiten Sie mit diesen Repräsentationen die folgenden Aufgaben:

(a) ▶ Geben Sie einen deterministischen Algorithmus A an, der bei Eingabe einer booleschen Formel φ für f mit n Variablen eine Belegung $(x_1,\ldots,x_n)$ berechnet, so dass $f(x_1,\ldots,x_n) = 1$ gilt, falls eine solche Belegung existiert, und andernfalls "nicht erfüllbar" ausgibt.

▶ Schätzen Sie die Worst-Case-Laufzeit Ihres Algorithmus in Abhängigkeit von der Anzahl der Variablen n ab. Sie können annehmen, dass das Auswerten einer Belegung Zeit α benötigt.

(b) ▶ Geben Sie einen deterministischen Algorithmus B an, der bei Eingabe einer Wahrheitstabelle T für f eine Belegung $(x_1,\ldots,x_n)$ berechnet, so dass $f(x_1,\ldots,x_n) = 1$ gilt, falls eine solche Belegung existiert, und andernfalls "nicht erfüllbar" ausgibt.

▶ Schätzen Sie die Worst-Case-Laufzeit Ihres Algorithmus in Abhängigkeit von der Länge der Wahrheitstabelle $|T|$ ab.

Lösungsvorschlag:

(a) Insgesamt gibt es 2^n verschiedene Belegungen in $\{0,1\}^n$. Unser Algorithmus testet alle 2^n Belegungen der Reihe nach durch. Dazu beginnt er mit der Belegung $(0,\ldots,0) \in \{0,1\}^n$ und hört bei der Belegung $(1,\ldots,1) \in \{0,1\}^n$ auf. Für jede Belegung $(x_1,\ldots,x_n)$ testet der Algorithmus, ob $\varphi(x_1,\ldots,x_n)$ wahr

ist. Hat der Algorithmus eine erfüllende Belegung gefunden, so gibt er diese aus und beendet das Testen. Hat der Algorithmus alle Belegungen getestet und keine erfüllende Belegung gefunden, so gibt er "nicht erfüllbar" aus.

Für das Auswerten einer Belegung wird Zeit α benötigt. Im schlimmsten Fall testet der Algorithmus alle 2^n Belegungen. Damit beträgt die Worst-Case-Laufzeit $\mathcal{O}(\alpha 2^n)$.

(b) Unser Algorithmus durchläuft von oben nach unten alle $|T|$ Zeilen der Wahrheitstabelle T. Für jede Zeile der Tabelle prüft der Algorithmus, ob in der Zelle mit dem Funktionswert für die entsprechende Belegung eine 1 steht. Ist dies für eine Zeile der Fall, so gibt der Algorithmus die Belegung der Zeile aus und ist fertig. Ist der Algorithmus am Ende der Tabelle angekommen, so gibt er "nicht erfüllbar" aus.

Die Wahrheitstabelle wird im schlimmsten Fall vollständig von oben nach unten durchlaufen, so dass die Wort-Case-Laufzeit $\mathcal{O}(|T|)$ beträgt.

Aufgabe 2 : Nichtdeterministisches Zeitkomplexitätsmaß

Betrachten Sie die folgende Sprache

$$\textsc{Composite}_{\textsc{unary}} = \{1^n \mid n \in \mathbb{N} \text{ ist keine Primzahl}\} \subseteq \{1\}^*.$$

Die 2-Band-NTM $N = (\Sigma, \Gamma, Z, \delta, z_0, \square, F)$ akzeptiert $\textsc{Composite}_{\textsc{unary}}$ mit $\Sigma = \{1\}$, $\Gamma = \{1, \widehat{1}, \square\}$, $Z = \{z_0, z_1, z_2, z_3, z_4, z_Y, z_N, z_S, z_A, z_R\}$, $F = F_A \cup F_R$, $F_A = \{z_A\}$, $F_R = \{z_R\}$ sowie δ:

δ	$(1,1)$	$(1,\square)$	$(\square,\square)$	$(1,\widehat{1})$
z_0		$(z_1,(1,\widehat{1}),R)$	$(z_R,(\square,\square),N)$	
z_1		$(z_1,(1,1),R)$	$(z_2,(\square,\square),L)$	
z_2	$(z_3,(1,\square),L)$			$(z_R,(1,\widehat{1}),N)$
z_3	$\{(z_3,(1,\square),L),\ (z_4,(1,1),L)\}$			$(z_R,(1,\widehat{1}),N)$
z_4	$(z_4,(1,1),L)$			$(z_S,(1,\widehat{1}),N)$
z_Y				$(z_A,(1,\widehat{1}),N)$
z_N				$(z_R,(1,\widehat{1}),N)$

Beide Köpfe bewegen sich stets in die selbe Richtung, so dass es genügt nur eine Richtung für beide Köpfe in δ anzugeben. N ist ausgestattet mit einem Eingabe– (1. Band) und einem Arbeitsband (2. Band) und besitzt eine Subroutine S, die mit einem Wechsel in den Zustand z_S aufgerufen wird. S überprüft in einem Berechnungsschritt ob die Zahl auf dem Arbeitsband die Zahl auf dem Eingabeband teilt. Falls ja, so wechselt N in den Zustand z_Y und andernfalls in z_N.

(a) ▶ Bestimmen Sie $NTime_N(1^6)$ und geben Sie alle akzeptierenden Konfigurationenfolgen an.

(b) ▶ Argumentieren Sie, weshalb N die Sprache $\text{COMPOSITE}_{\text{UNARY}}$ für eine Eingabe $x \in \{1\}^*$ in Zeit $2|x| + 3$ akzeptiert. Sie können dabei $L(N) = \text{COMPOSITE}_{\text{UNARY}}$ voraussetzen.

Lösungsvorschlag:

(a) Wir geben alle akzeptierenden Konfigurationenfolgen von N für die Eingabe $x = 1^6$ an. Dazu geben wir erst den Anteil aller akzeptierenden Konfigurationenfolgen an, den diese gemeinsam haben

$$z_0(1^6, \square) \vdash_N (1, \widehat{1})z_1(1^5, \square) \vdash_N (1^2, \widehat{1}1)z_1(1^4, \square) \vdash_N (1^3, \widehat{1}1^2)z_1(1^3, \square) \vdash_N$$
$$(1^4, \widehat{1}1^3)z_1(1^2, \square) \vdash_N (1^5, \widehat{1}1^4)z_1(1, \square) \vdash_N (1^6, \widehat{1}1^5)z_1(\square, \square) \vdash_N (1^5, \widehat{1}1^4)z_2(1, 1) \vdash_N$$
$$(1^4, \widehat{1}1^3)z_3(1^2, 1) \vdash_N (1^3, \widehat{1}1^2)z_3(1^3, 1) \vdash_N (1^2, \widehat{1}1)z_3(1^4, 1) \vdash_N \ldots.$$

Ab dieser Stelle unterscheiden sich die akzeptierenden Konfigurationenfolgen wie folgt:

(i) zum Testen des Teilers 3:

$$\ldots (1, \widehat{1})z_4(1^5, 1^2) \vdash_N z_4(1^6, \widehat{1}1^2) \vdash_N z_S(1^6, \widehat{1}1^2) \vdash_N z_Y(1^6, \widehat{1}1^2) \vdash_N z_A(1^6, \widehat{1}1^2),$$

(ii) zum Testen des Teilers 2:

$$\ldots (1, \widehat{1})z_3(1^5, 1) \vdash_N z_4(1^6, \widehat{1}1) \vdash_N z_S(1^6, \widehat{1}1) \vdash_N z_Y(1^6, \widehat{1}1) \vdash_N z_A(1^6, \widehat{1}1).$$

Beide akzeptierenden Pfade haben die gleiche Länge. Da die einzigen Teiler von 6, die größer als 1 sind, 2 und 3 sind, gibt es keine weiteren akzeptierenden Konfigurationenfolgen und es folgt $NTime_N(1^6) = 15$.

(b) Für eine Eingabe $x \in \{1\}^*$ arbeitet N wie folgt. N bewegt den Kopf einmal von links nach rechts über die Eingabe, dies benötigt $|x|$ Schritte. Anschließend bewegt N den Kopf von rechts nach links über die Eingabe zurück und verzweigt sich in die nichtdeterministischen Berechnungen, dies benötigt ebenfalls $|x|$ Schritte. Das letzte Aufrufen von S sowie das Auswerten des Ergebnisses benötigt weitere 3 Schritte. Insgesamt benötigt N also $2|x| + 3$ Arbeitsschritte.

Aufgabe 3 : Berechnungsbäume

Sei M eine NTM mit Eingabealphabet Σ die immer anhält und sei $x \in \Sigma^*$. Der *endliche Berechnungsbaum* T von M bei Eingabe x ist ein gerichteter Baum, dessen Knoten den Konfigurationen in der Berechnung $M(x)$ entsprechen. Die Wurzel von T stellt die Startkonfiguration von $M(x)$ dar. In T existiert eine Kante von Knoten a nach Knoten b genau dann, wenn b eine Nachfolgekonfiguration von a in der Berechnung $M(x)$ ist, also wenn man ausgehend von Konfiguration a Konfiguration b in einem Schritt erreicht.

Sei $x \in L(M)$ und T der Berechnungsbaum für $M(x)$. Wir definieren die *Höhe* von T als die Länge eines längsten Pfades von der Wurzel von T zu einem Blatt in T. Weiter definieren wir die *Länge eines Pfades* p in T als die Anzahl der Kanten in p.

(a) ▶ Zeigen Sie:

Wenn T eine Höhe kleiner gleich $h \in \mathbb{N}$ hat, dann gilt $NTime_M(x) \leq h$.

(b) ▶ Beweisen oder widerlegen Sie die Rückrichtung der Aussage in (a).

Lösungsvorschlag:

(a) Zu Beginn stellen wir fest, dass per Voraussetzung $x \in L(M)$ gilt und daher $NTime_M(x)$ definiert ist. Nun beweisen wir die Aussage per Widerspruch. Angenommen T hat eine Höhe kleiner gleich h aber es gilt $NTime_M(x) > h$. Aus $NTime_M(x) > h$ folgt, dass die Länge eines kürzesten, akzeptierenden Berechnungspfades von M auf Eingabe x mindestens $h + 1$ beträgt. Dies ist aber ein Widerspruch dazu, dass die Höhe von T höchstens h beträgt, da die Höhe von T der Länge eines längsten Berechnungspfades von M auf Eingabe x entspricht und damit eine obere Schranke für die Länge aller endlichen Berechnungspfade von M auf Eingabe x darstellt. Folglich kann $NTime_M(x) > h$ nicht gelten und es muss $NTime_M(x) \leq h$ gelten.

(b) Im Allgemeinen gilt die Aussage nicht. Wir überlegen uns, dass

$$NTime_M(x) \leq h$$

die Länge eines *kürzesten*, akzeptierenden Berechnungspfades beschränkt. Es kann jedoch längere, endliche Berechnungspfade geben, so dass die Höhe des Berechnungsbaums T echt größer als h ist.

Aufgabe 4 : Abschlusseigenschaften

▶ Zeigen Sie:

DSPACE(s) *ist unter (i) Vereinigung und (ii) Schnittbildung abgeschlossen.*

Lösungsvorschlag:

(i) Seien $A, B \in \text{DSPACE}(s)$ zwei gegebene Sprachen. Dann existieren zwei DTMs M_A, M_B mit $L(M_A) = A, L(M_B) = B$ und für jedes $n \in \mathbb{N}$ gilt: $space_{M_A}(n) \leq s(n)$ sowie $space_{M_B}(n) \leq s(n)$.

Um zu zeigen, dass DSPACE(s) abgeschlossen unter Vereinigung ist, müssen wir zeigen, dass auch $A \cup B \in \text{DSPACE}(s)$ gilt. Dazu definieren wir eine neue DTM M', für die $L(M') = A \cup B$ gelten soll. M' funktioniert wie folgt. Für eine Eingabe x simuliert M' zuerst die Berechnung der Maschine M_A auf Eingabe x.

Falls M_A die Eingabe x akzeptiert, so akzeptiert auch M' die Eingabe. Falls M_A die Eingabe x nicht akzeptiert, so simuliert M' als nächstes die Berechnung von M_B auf Eingabe x. Falls M_B die Eingabe x akzeptiert, so akzeptiert auch M' die Eingabe, andernfalls lehnt M' die Eingabe x ab. Dann gilt $L(M') = A \cup B$.

Um $M_A(x)$ zu simulieren benötigt M' höchstens einen Platz von $s(|x|)$. Analog benötigt M', um $M_B(x)$ zu simulieren, höchstens einen Platz von $s(|x|)$. Insgesamt braucht M' also bei Wiederverwendung des genutzten Platzes höchstens einen Platz von $s(|x|)$, so dass $space_{M'}(n) \leq s(n)$ gilt und damit $A \cup B \in \text{DSPACE}(s)$ folgt.

(ii) Seien $A, B \in \text{DSPACE}(s)$ wie in (i). Um zu zeigen, dass $\text{DSPACE}(s)$ unter Schnittbildung abgeschlossen ist, müssen wir zeigen, dass $A \cap B \in \text{DSPACE}(s)$ gilt. Dazu definieren wir eine neue DTM M', für die $L(M') = A \cap B$ gelten soll. M' funktioniert wie folgt. Für eine Eingabe x simuliert M' zuerst die Berechnung von M_A auf Eingabe x. Falls M_A ablehnt, so lehnt M' ebenfalls ab. Falls M_A akzeptiert, so simuliert M' im nächsten Schritt die Berechnung von M_B auf Eingabe x. Falls M_B ablehnt, so lehnt M' auch ab, andernfalls akzeptiert M' die Eingabe x. Dann gilt $L(M') = A \cap B$.

Um $M_A(x)$ zu simulieren benötigt M' höchstens einen Platz von $s(|x|)$. Analog benötigt M' höchstens einen Platz von $s(|x|)$ um $M_B(x)$ zu simulieren. Insgesamt benötigt M' also höchstens einen Platz von $s(|x|)$, so dass $space_{M'}(n) \leq s(n)$ gilt und damit $A \cap B \in \text{DSPACE}(s)$ folgt.

Damit gilt formal ausgedrückt:

(a) $A, B \in \text{DSPACE}(s) \Rightarrow A \cup B \in \text{DSPACE}(s)$ und

(b) $A, B \in \text{DSPACE}(s) \Rightarrow A \cap B \in \text{DSPACE}(s)$.