Theoretische Informatik Serie 9

Benjamin Simmonds Dario Näpfer Fabian Bösiger

Aufgabe 24

(a)

Wir konstruieren eine eine MTM A wobei gilt $L(A) = L(M)$ und die Berechnungen von M simuliert. Wir können laut Satz 6.6 annehmen, dass für jedes Wort $w \in L(M)$ es nur eine eindeutige Konfiguration gibt. Somit müssen wir nur erkennen ob $C_{accept(w)}$ von C_{start} erreichbar ist. Wir wissen, dass die kürzeste akzeptierende Berechnung von M auf w höchstens Länge $n^2 * c$ besitzt für eine Konstante c, da $Time_M(n) \in O(n^2)$.

Wir werden die Prozedur $REACHABLE$ vom Beweis von Savitch benutzen um zu erkennen, ob $C_{accept(w)}$ von C_{start} in $n^2 * c$ Schritten erreichbar ist. A muss bei der Durchführung von $REACHABLE$ höchstens $log(n^2*c) = 2log(n)+log(c)$ Konfigurationen auf einmal speichern, weil die Anzahl der verschachtelten Rekursionsaufrufe höchstens so gross ist.

Laut Aufgabenstellung kann jede innere Konfiguration einer Berechnung in $O(n) = n * d$ Platz gespeichert werden. Die Prozedur $REACHABLE$ muss höchstens $O(log(n))$ Konfigurationen der Länge $O(n)$ aufs Mal speichern. Somit können wir mit der analogen Argumentation wie beim Beweis des Satzes von Savitch den Platzbedarf von A in $O(n * log(n))$ begründen.

(b)

Für jede Sprache $L \in NSPACE(f(n)) \cap NTIME(f(n)^k) \Leftrightarrow L \in NSPACE(f(n))$ und $L \in NTIME(f(n)^k)$. Somit gibt es eine nichtdeterministische MTM M_1 mit $L(M_1) = L$ und $Space_{M_1}(n) \in O(f(n))$ sowie auch eine nichtdeterministische MTM M_2 mit $L(M_2) = L$ und $Time_{M_2}(n) \in O((f(n))^k)$.

Es ist somit möglich, dass es eine MTM gibt, die L mit kleiner Platzkomplexität entscheidet, und eine andere MTM, die L mit geringer Zeitkomplexität entscheidet. Wir können aber nicht einen Beweis wie in a) führen, denn dafür bräuchten wir eine nichtdeterministische MTM, die beide Schranken für Platz und Zeit einhält.

Aufgabe 25

(a)

Wir zeigen $VC \leq_p SCP$. Zuerst modellieren wir die Eingabe (G, k) für VC um zu einer Eingabe für SCP: Wir wählen (E, S_G, k), wobei E_v die Menge der Kanten ist, die zu v inzident sind, also $E_v = \{e \in E \mid v \textit{ ist inzident zu } e\}$. S_G definieren wir als $S_G = \{E_v | v \in V\}$. Diese Ummodellierung können wir in polynomieller Zeit durchführen.

Wir zeigen $(G, k) \in VC \Leftrightarrow (E_v, S_G, k) \in SCP$:

Sei $(G, k) \in VC$. Dementsprechend existiert eine Knotenmenge $M = \{v_1, v_2, ..., v_k\}$, die alle Kanten überdeckt. Da die überdeckten Kanten der Vereinigung aller E_{v_i} für $v_i \in M$ entsprechen, existiert ein Set-Cover mit Grösse k, weshalb gilt, dass $(E_v, S_G, k) \in SCP$.

Sei $(E_v, S_G, k) \in SCP$. Dann gibt es eine Teilmenge C von S_G mit Kardinalität k, dessen Vereinigung E ergibt. Das heisst, dass diejenigen k Knoten, die nach der Ummodellierung auf Mengen aus C entsprechen, alle Kanten aus E überdecken und somit ein Vertex-Cover der Grösse k bilden.

(b)

Wir zeigen $SCP \leq_p DS$. Zuerst modellieren wir die Eingabe $X = \{x_1, x_2, ..., x_n\}$ und $S = S_1, S_2, ..., S_m$ und eine natürliche Zahl k für SCP zu einer Eingabe für DS: Wir erstellen einen Graphen $G = (V, E)$. Wir modellieren alle $x_i \in X$ und alle $S_i \in S$ als Knoten. Wir definieren folgende Kanten für alle $x_i \in X$ und alle $S_i \in S$: $e = \{x_i, s_j\} \Leftrightarrow x_i \; inS_j$. Zudem erstellen wir zusätzliche Kanten, so dass die Knoten S_j untereinander einen vollständigen Graph bilden. Diese Modellierung können wir in polynomieller Zeit durchführen.

Sei $(E_v, S_G, k) \in SCP$. Dann gibt es eine Teilmenge C von S_G mit Kardinalität k, dessen Vereinigung E ergibt. Die den Elementen aus C entsprechenden Knoten aus V bilden somit ein Dominating-Set D der Grösse k, da jeder Knoten, der einem x_i entspricht, mit einem Knoten S_j verbunden ist und jedes S_k mit einem S_j verbunden ist, das Teil von C und somit D ist.

Sei D ein Dominating-Set der Grösse k. Wenn D nur aus Knoten S_j besteht, gilt offensichtlich, dass die entsprechenden S_j zusammen ein Set-Cover bilden, da alle x_i zu mindestens einem dieser S_j adjazent sind. Wenn aber D auch noch aus anderen Knoten besteht, können wir einfach den Knoten dieses x_i mit einem adjazenten Knoten S_j vertauschen (bemerke, dass somit beide dominiert bleiben und dass alle Nachbarn von x trotz dem Vertauschen dominiert bleiben).

Aufgabe 26

Wir zeigen $MonoSAT \in NP$. Dazu beschreiben wir einen Verifizierer A für $MonoSAT$. Wir verwenden den Verifizierer B für SAT als Teilprogramm. Für jede Eingabe (Φ, x)

überprüft A zunächst, ob alle Klauseln in Φ monoton sind. Falls dies nicht der Fall ist, gibt A *falsch* aus. Ansonsten übergibt A die Eingabe (Φ, x) in den Verifizierer B, der überprüft, ob Φ durch den Zeugen x verifiziert werden kann. A gibt abschliessend die Ausgabe von B aus.

Wir zeigen anschliessend, dass $SAT \leq_p MonoSAT$. Sei $F = F_1 \wedge F_2 \wedge ... \wedge F_m$ eine Formel in KNF über einer Menge Boole'scher Variablen $\{x_1, ..., x_n\}$. Wir Konstruieren die Formel $C = C_1 \wedge C_2 \wedge ... \wedge C_m$ in KNF, für die alle Klauseln monoton sind, so dass $F \in SAT \Leftrightarrow C \in MonoSAT$.

Die polynomielle Reduktion führen wir für jede der Klauseln $F_1, ..., F_m$, einzeln wie folgt durch. Falls F_i monoton ist, können wir $C_i = F_i$ übernehmen. Falls F_i nicht monoton ist, konstruieren wir zwei neue Klauseln $B_{i,0}$ und B_{i, 1}, wobei wir alle positiven Variablen in der Klausel F_i in $B_{i,0} = (x_k \vee ... \vee x_l \vee y_i)$ einfügen, und alle negativen in $B_{i,1} = (\bar{x}_m \vee ... \vee \bar{x}_n \vee \bar{y}_i)$. Anschliessend erstellen wir die neue Doppelklausel $C_i = B_{i,0} \wedge B_{i,1}$.

Für $F_i = (\bar{x}_1 \vee x_2 \vee x_3)$ erhalten wir zum Beispiel $C_i = (x_2 \vee x_3 \vee y_i) \wedge (\bar{x}_1 \vee \bar{y}_i)$.

Um zu zeigen, dass F genau dann erfüllbar ist, wenn C erfüllbar ist, reicht es, die folgende Behauptung aus dem Buch zu beweisen.

> Eine Belegung φ der Variablen aus $\{x_1, x_2, ..., x_n\}$ erfüllt $F_i \Leftrightarrow$ Es existiert eine Erweiterung φ' von φ auf $\{x_1, y_1, x_2, y_2, ..., x_n, y_n\}$, die C_i erfüllt.

Wir beweisen im Folgenden beide Richtungen.

"$\Rightarrow$": Sei φ eine Belegung der Variablen in $\{x_1, x_2, ..., x_n\}$, so dass $\varphi(F_i) = 1$. Also existiert ein x_j in der i-ten Klausel mit $\varphi(x_j) = 1$, falls x_j positiv ist, beziehungsweise $\varphi(\bar{x}_j) = 1$, falls x_j negativ ist. Wir nehmen φ', so dass $\varphi'(x_i) = \varphi(x_i)$, und $\varphi'(y_r) = 0$, falls x_j in der Klausel $B_{r,0}$ ist, $\varphi'(y_r) = 1$, falls $\bar{x}_j$ in der Klausel $B_{r,1}$ ist. Falls die Aufteilung in $B_{r,1}$ und $B_{r,0}$ nicht erfolgt ist, also F_r bereits monoton war, kommt y_r nicht in C vor und kann deshalb frei gewählt werden. Nach Annahme erfüllt die Belegung φ die Anforderungen von F_i für alle $i \in \{1, ..., n\}$. Falls F_i monoton ist, haben wir $C_i = F_i$ und somit ist $\varphi'(C_i) = 1$. Falls F_i nicht monoton ist, haben wir entweder $\varphi'(B_{r,0}) = 1$, weil $\varphi'(x_j) = 1$ und $\varphi'(B_{r,1}) = 1$, weil nach unserer Wahl dann $\varphi'(\bar{y}_i) = 1$, oder wir haben $\varphi'(B_{r,1}) = 1$, weil $\varphi(\bar{x}_j) = 1$ und $\varphi'(B_{r,0}) = 1$, weil nach unserer Wahl dann $\varphi'(y_i) = 1$. Somit gilt $\varphi'(C_i) = \varphi'(B_{i,0} \wedge B_{i,1}) = 1$.

"$\Leftarrow$": Sei φ eine Belegung, so dass $\varphi(F_i) = 0$. Wir beweisen, dass keine Erweiterung φ' von φ existiert, so dass $\varphi'(C_i) = 1$. $\varphi(F_i) = 0$ impliziert, dass für alle Variablen x_w in der Klausel i gilt, dass $\varphi(x_w) = 0$, falls x_w positiv ist, beziehungsweise $\varphi(\bar{x}_w) = 0$, falls x_w negativ ist. Falls F_i monoton ist, gilt offensichtlich $\varphi'(C_i) = \varphi(F_i) = 0$. Falls F_i nicht monoton ist, wissen wir, dass x_w in der Klausel $B_{i,0}$ ist, falls x_w positiv ist, und dass $\varphi'(x_w) = 0$, oder x_w landet in der Klausel $B_{i,1}$, falls x_w negativ ist, und $\varphi'(\bar{x}_w) = 0$. Somit können wir y_i nie so wählen, dass gleichzeitig $\varphi'(B_{i,0}) = 1$ und $\varphi'(B_{i,0}) = 1$. Da $C_i = B_{i,0} \wedge B_{i,1}$, muss deshalb $\varphi'(C_i) = 0$ sein.

Da $MonoSAT \in NP$ und $SAT \leq_p MonoSAT$, ist $MonoSAT$ NP-vollständig.