GTI Gedächtnisprotokoll

Aufgabe 1: [Potenzmengenautomat]
Gegeben war ein recht einfacher NFA (ohne epsilon-Übergänge). Man sollte den NFA durch Potenzmengenkonstruktion in einen äquivalenten DFA überführen

Aufgabe 2: [Minimierungsalgorithmus]
Gegeben war ein etwas komplexerer DFA mit 3 akzeptierenden Zuständen. Der DFA sollte über den Minimierungsalgorithmus minimiert werden und der minimale Automat angegeben werden

2 Zustände waren vom Startzustand aus nicht erreichbar und konnten direkt gestrichen werden.
Zum Schluss ließen sich einmal 2 akzeptierende und 2 nicht-akzeptierende zusammenfassen

Aufgabe 3: [Kontextfreie Grammatiken]
a) Gegeben war eine kontextfreie Grammatik ohne nutzlose Variablen. Man sollte die Schritte CNF3 und CNF4 aus der Chomsky-Normalform anwenden und erklären und die Grammatik nach CNF3 angeben

b) Gegeben war eine andere kontextfreie Grammatik. Man sollte zeigen, dass die Grammatik beide Bedingungen an eine LL(1)-Grammatik nicht erfüllte (die Definition für FIRST- und FOLLOW-Mengen waren auf dem Zettel angegeben, aber *nicht* die Bedingungen für LL(1)-Grammatiken)

c) Man sollte eine möglichst einfache Grammatik angeben, sodass eine Variable in CNF5 (beim Entfernen der Einzelregeln) nicht erreichbar wird

Beispiel: $G = (\{S, A\}, \{a\}, S, P)$ mit Regeln:
$S \rightarrow A$
$A \rightarrow a$

$S \rightarrow A$ wird in CNF5 durch $S \rightarrow a$ ersetzt, A wird nicht erreichbar

Aufgabe 4: [Kategorisierung von Sprachen]
Gegeben waren 3 Sprachen:

a) Die Sprachen sollten eingeordnet werden, und
b) die Einordnung begründet werden (bei Angabe von Automaten / Grammatiken / regulären Ausdrücken sollte zusätzlich die Bedeutung / Konstruktion erklärt werden)

L_1: nicht kontextfrei (Pumpinglemma mit $w = a^n c^n a^n$)

*L_2: regulär, möglicher Ausdruck: aacc*aa + abcc*ba + bacc*ab + bbcc*bb*

L_3: kontextfrei (PDA: liest erst u ein, dann über Teilautomat mit 3 Zuständen eine durch drei teilbare Anzahl an cs, gleicht danach Reststring mit Kellerinhalt ($=u^R$) ab)
aber nicht regulär (Pumpinglemma mit $w=a^n c^3 a^n$)

Diese Aufgabe war mit Abstand diejenige, welche die meisten Punkte gab

Aufgabe 5: [Entscheidungs- vs. Optimierungsprobleme]
Gegeben waren drei verwandte Probleme:

COMMONSUBSEQUENCE:
Eingabe: Wörter w1, ..., wn; Zahl k
Frage: existiert eine gemeinsame Teilsequenz u für alle Wörter mit |u| >= k?

COMMONSUBSEQUENCEW:
Eingabe Wörter w1, ..., wn
Frage: was ist die Länge der längsten gemeinsamen Teilsequenz?

COMMONSUBSEQUENCEO:
Eingabe: Wörter w1, ..., wn
Frage: was ist die längste gemeinsame Teilsequenz

a) Man sollte zeigen, dass COMMONSUBSEQUENCEW polynomiell berechenbar ist, wenn COMMONSUBSEQUENCE polynomiell berechnbar ist

*Binäre Suche im Intervall [0, |w1|], jeweils Anwenden eines Lösungsalgorithmus für COMMONSUBSEQUENCE. Laufzeit ist polynomiell: O(log |w1| * T(A))*

b) Man sollte zeigen, dass COMMONSUBSEQUENCEO ebenfalls polynomiell berechenbar ist:

Sequentiell über die Zeichen von w1 iterieren, aktuelles Zeichen entfernen. Wenn sich Wert für COMMONSUBSEQUENCEO ändert, ist das Zeichen Teil einer längsten gemeinsamen Teilfolge

Aufgabe 6: [Polynomielle Reduktion]
Gegeben waren zwei Probleme:

CLIQUEÜBERDECKUNG:

Eingabe: Graph G = (V,E), Zahl k
Frage: kann man V in k Partitionen unterteilen, sodass jede Partition eine Clique bildet?

COL
Eingabe: Graph G = (V,E), Zahl k
Frage: kann man eine k-Färbung für G finden? (Also: lässt sich eine Funktion c : V -> {1,...,k} finden, sodass c(v) = c(u) => (v,u) nicht in E)

Definiert war außerdem das inverse eines Graphen: !G = (V, !E), wobei !E = V x V \ E. Der Graph !G hat also eine Kante zwischen 2 Knoten genau dann, wenn G keine Kante zwischen den Knoten hat

Zusätzlich waren 2 Funktionen gegeben:
$f_1(G,k) = (!G, k)$
$f_2(G,k) = (!G, |V| - k)$

a) Man sollte für einen gegebenen Graphen H begründen, ob es eine Cliquenüberdeckung mit k=2 für den Graphen gibt
b) Man sollte für den gleichen Graphen H begründen, für welche der angegebenen Funktionen gilt: $f_i(H,2)$ ist in COL
c) Man sollte begründen, welche der beiden Funktionen keine Reduktion von CLIQUENÜBERDECKUNG auf COL ist.

Mit a) und b) hatte man direkt ein Gegenbeispiel für f_2

Man sollte für die andere Funktion formal zeigen, dass gilt:
(G,k) in CLIQUENÜBERDEKCKUNG => f(G,k) in COL
Die anderen Eigenschaften für polynomielle Reduktionen sollten in ein paar Sätzen begründet werden

Grundidee: (G,k) in CLIQUENÜBERDECKUNG => es gibt eine Partitionierung V1, ..., Vk mit u,v in Vi => (u,v) in E => (u,v) nicht in !E => c(v) = i <=> v in Vi ist eine gültige k-Färbung => f(G,k) ist in COL

d) Wenn COL NP-vollständig ist, was folgt (im Hinblick auf c)) damit für CLIQUENÜBERDECKUNG?

CLIQUENÜBERDECKUNG liegt in NP.

Aufgabe 7: [2-Kopf-Turingmaschinen]
Gegeben war eine recht lange Definition von 1-String-Turingmaschinen, die zwei Lese-Schreibe-Köpfe haben, die seperat bewegt und beschrieben werden können. Dazu gab es

eine erweiterte Definition von Konfigurationen: (q, (w, z1, z2)) wobei q der aktuelle Zustand, w der aktuelle String, z1 die Position des ersten Kopfes und z2 die Position des 2. Kopfes ist. Die Transitionsfunktion hängt zusätzlich nun von Paaren von gelesenen Zeichen ab und hat entsprechend jeweils Paare von Reichtungsänderungen und geschriebenen Zeichen (wenn beide Köpfe auf der gleichen Position stehen, darf nur der erste Kopf schreiben).

a) Angabe einer Startkonfiguration für einen gegebenen String w

(s, (w, 0, 0))

Außerdem war eine Transitionsfunktion für eine 2-Kopf-Turingmaschine gegeben, mit deren Hilfe man 3 Nachfolgekonfigurationen für eine gegebene Konfiguration angeben sollte

b) Man sollte begründen, warum jede von einer 2-Kopf-Turingmaschine berechenbaren Funktion berechenbar ist

Grundidee: Die Turingmaschine wird von einer 2-String-1-Kopf-Turingmaschine simuliert, die sich auf den 2. String die Positionen der einzelnen Köpfe notiert und in jedem Schritt der Ursprungs-Turingmaschine sich zu jeder Position einzeln bewegt und die Positionen entsprechend überschreibt sowie die Kopfposition im 2. String anpasst. Außerdem: Church-Turing-These

Aufgabe 8: Quizfragen

1.: Ein gegebener regulärer Ausdruck a lässt sich in O(|a|) in einen PDA verwandeln

Richtig: ein RE lässt sich in O(|a|) in ein epsilon-NFA verwandeln. Ein epsilon-NFA ist im Prinzip ein PDA, der seinen Keller ignoriert

2.: Jede Grammatik in Chomsky-Normalform ist eindeutig

Falsch: jede Grammatik von inhärent mehrdeutigen Sprachen lassen sich auch in eine Chomsky-Normalform überführen

3.: Kontextfreie Sprachen haben unendlich viele Nerode-Äquivalenzklassen

Falsch: jede kontextfreie Sprache, die regulär ist, hat endlich viele Nerode-Äquivalenzklassen (Satz von Myhill und Nerode)

4.: Für jeden DFA gibt es einen epsilon-NFA mit echt weniger Zuständen

Falsch: der minimale DFA für die leere Menge hat nur einen Zustand. Kein endlicher

Automat kann weniger Zustände haben (dieser müsste 0 Zustände besitzen)

5.: Das Komplement von CFG-SCHNITT ist semi-entscheidbar

Falsch: CFG-Schnitt ist semi-entscheidbar, aber nicht entscheidbar. Somit kann das Komplement von CFG-Schnitt nicht semi-entscheidbar sein, da ansonsten CFG-Schnitt berechenbar wäre.