8. Aufgabenblatt zur Vorlesung

Grundlagen der Theoretischen Informatik, SoSe 2008

(Dr. Frank Hoffmann)

Musterlösung von Benjamin Bortfeldt, Manuel Jain

Aufgabe 1 Entscheidungs– vs. Optimierungsproblem (2 Punkte)

Angenommen ein Algorithmus A könnte in polynomieller Zeit das Entscheidungsproblem CLIQUE = $\{< G, k >|$ Graph G enthält eine Clique der Grösse $k\}$ lösen. Geben Sie einen Algorithmus an, der das dazugehörige Optimierungsproblem in polynomieller Zeit löst und dazu A verwendet. Dabei sei ein Graph G gegeben und es ist die Anzahl der Knoten in einer größte Clique in diesem Graphen gesucht.

Lösung: Sei $G = (V, E)$ mit $|V| = n$ gegeben. Der Graph G kann als maximale Clique nun eine der Größe n haben, das ist dann der Fall, wenn G vollständig ist. Daher wird das gesuchte maximale k aus der Menge $\{1, 2, \ldots, n\}$ kommen. Die einfachste Lösung ist sicherlich durch diese Liste zu iterieren und für jedes k den Algorithmus A anzuwenden. Falls zum ersten Mal für ein k keine Clique in G gefunden wird, handelt es sich bei der Clique der Größe (k-1) um die maximale Cliquengöße. Falls für $k = n$ eine Clique gefunden wird, hat die größte Clique die Größe n.
Falls die maximale Clique die Größe n hat, muss der Algorithmus A n-mal durchgeführt werden. Da nach Voraussetzung für die Laufzeit von A gilt, dass sie durch ein Polynom abschätzbar ist (also in $O(n^i)$ liegt für ein konstantes i), befindet sich die Laufzeit des beschriebenen Algorithmus in $O(n^{i+1})$, also auch polynomielle Laufzeit. Etwas geschickter kann man vorgehen, wenn man den möglichen Bereich mittels Binärsuche durchsucht, dann ergibt sich in der Laufzeit ein zusätzlicher Faktor von $\log n$ und nicht n für die eben beschriebene lineare Suche.

Aufgabe 2 Polynomielle Reduktion I (4 Punkte)

Zeigen Sie, dass gilt: HAMILTON $\leq_p$ SUBGRAPHISO.
Dabei ist HAMILTON das Entscheidungsproblem, ob ein ungerichteter Graph einen Kreis enthält, auf dem alle Knoten genau einmal liegen, und SUBGRAPHISO ist die Frage, ob für ein Paar $< H, G >$ von ungerichteten Graphen gilt, dass es in G einen Untergraphen H' gibt, so dass H und H' isomorph sind.

Lösung: Wurde im Tutorium besprochen.

Aufgabe 3 Polynomielle Reduktion II (4 Punkte)

Zeigen Sie, dass das Problem DOUBLE-SAT NP–vollständig ist. Benutzen Sie zur Reduktion das Problem SAT. DOUBLE-SAT ist die Sprache aller Booleschen KNF–Formeln Φ, die mindestens zwei verschiedene erfüllende Belegungen haben.

Lösung: Wurde im Tutorium besprochen.

Aufgabe 4 Vom dfa zur Typ3–Grammatik (3 Punkte)

Zeichnen Sie das Zustandsdiagramm eines dfa, der über dem Alphabet $\{a, b\}$ alle Wörter erkennt, die nicht *aab* als Teilwort enthalten. Wandeln Sie den dfa in eine Typ3–Grammatik um.

Lösung: Wurde im Tutorium besprochen.

Aufgabe 5 Abgeschlossenheit mal anders (3 Punkte)

Beweisen Sie die Abgeschlossenheit der Klasse der regulären Sprachen gegenüber Vereinigung, Konkatenation und Kleeneschen Abschluss direkt mittels regulärer Grammatiken.

Lösung: Seien $G_1 = (V_1, \Sigma, P_1, S_1)$ mit $L_1 = L(G_1)$ und $G_2 = (V_2, \Sigma, P_2, S_2)$ $L_2 = L(G_2)$ Typ3-Grammatiken, die demnach reguläre Sprachen erzeugen.

Vereinigung: Zu zeigen: $L_1 \cup L_2$ regulär.
Ein gegebenes Wort muss also mit G_1 oder G_2 produzierbar sein. Erstelle dazu eine neue Grammatik $G_\cup = (V_1 \cup V_2, \Sigma, P_1 \cup P_2, S_\cup)$ und führe eine neue Produktion ein:

$$S_\cup \to S_1 | S_2$$

Durch die neue Produktion, kann man zu Beginn entscheiden, ob man das Wort mit G_1 oder G_2 herleitet und das ist genau das, was die Vereinigung ausmacht.

Konkatenation: Zu zeigen: $L_1 \circ L_2$ regulär:
Hier müssen wir nur G_1 geringfügig modifizieren:
Alle Produktionen haben die Form

$$A \to aB \text{ oder } \underbrace{A \to a}_{(1)}$$

wobei A, B Variablen und $a \in \Sigma \cup \{\epsilon\}$. Produktionen der Form (1) in G_1 werden nun folgendermaßen verändert:

$$A \to a \mapsto A \to aS_2$$

S_1, die Startvariable von G_1 ist auch Startvariable der neuen Grammatik.Somit muss ich nach dem Erzeugen eines Wortes aus L_1 ein Wort aus L_2 erzeugen und genau das war gefordert.

Kleene-Stern: Zu zeigen: L_1^* regulär:
Hier kann man ähnlich wie bei der Konkatenation vorgehen, man ersetzt folgendermaßen:

$$A \to a \mapsto A \to aS_1$$

somit können wir beliebig viele Wörter aus L_1 aneinanderhängen. Da in L_1^* aber auf jeden Fall auch das leere Wort enthalten ist, selbst wenn es in L_1 nicht enthalten ist, muss man noch eine weitere Produktion

$$S_1 \to \epsilon$$

einführen. Damit sichern wir auch, dass Ableitungen terminieren.

Aufgabe 6 Grammatiken (4 Punkte)

Geben Sie Grammatiken für die folgenden Sprache über $\Sigma = \{a, b, c\}$ an.

(a) $L = \{a^n b^n c^m \mid n, m \geq 1\}$

(b) $L' = \{a^j b^k c^l \mid j \geq k \geq l \geq 1\}$

Begründen Sie, dass damit L und L' erzeugt werden. Von welchem Chomsky–Typ sind Ihre Grammatiken?

Lösung:

(a) Die Produktionen um L zu erzeugen, können wie folgt aussehen:

$$S \to aAbCc \quad (1)$$
$$A \to aAb|\epsilon \quad (2)$$
$$C \to cC|\epsilon \quad (3)$$

Durch die Produktion von S ist sichergestellt, dass a,b und c mindestens einmal vorkommen. Die Produktion (2) erlaubt es, das Wort mit a's und b's aufzufüllen und stellt dabei sicher, dass immer gleichviele a's und b's da sind.
Die Produktion (3) ermöglicht es das Wort mit beliebig vielen c's aufzufüllen. In Produktion (1) ist bereits sichergestellt, dass die Reihenfolge eingehalten wird. Es handelt sich hierbei um eine Typ2 - Grammatik, da immer von genau einer Variablen abgeleitet wird.

(b) Die Produktionen, um L' zu erzeugen, können wie folgt aussehen.

$$S \to ABc|ABSc \quad (1)$$
$$BA \to AB \quad (2)$$
$$AB \to AABB \quad (3)$$
$$A \to AA \quad (4)$$
$$Bc \to bc \quad (5)$$
$$Bb \to bb \quad (6)$$
$$Ab \to ab \quad (7)$$
$$Aa \to aa \quad (8)$$

Hierbei handelt es sich fast(!) um eine Grammatik von Typ1, also kontextsensitiv. Auf der linken Seite steht zum Teil mehr als eine Variable, das ist ok, und teilweise Terminalsymbole und Variable gemischt, letzteres ist nicht erlaubt. Des weiteren ist die rechte Seite immer mindestens so groß wie die linke und auch das wird von einer Typ1 - Grammatik gefordert.Man beachte, dass nach Anwendung mehrerer (1)–Regeln, die A's und B's nicht in der gewünschten Reihenfolge stehen. Dann kann und muss Regel (2) angewendet werden, anders kommt man auch nicht zu Terminalsymbolen.
Jetzt kann man durch Einführung von dummy–Variablen H_a, H_b, H_c das Auftreten der Terminalsymbole auf der linken Seite verhindern und man erhält eine kontextsensitive Grammatik, mit der die Terminalzeichen in einem Wort $w \in L'$ von rechts nach links erzeugt werden.

$$S \to ABH_c|ABSH_c \tag{1}$$
$$BA \to AB \tag{2}$$
$$AB \to AABB \tag{3}$$
$$A \to AA \tag{4}$$
$$H_c \to c \tag{5}$$
$$BH_c \to H_bc \tag{6}$$
$$BH_b \to H_bb \tag{7}$$
$$AH_b \to H_ab \tag{8}$$
$$AH_a \to H_aa \tag{9}$$
$$H_a \to a \tag{10}$$