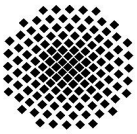
Universität Stuttgart
Fakultät 5, Institut FMI
EfidI 1, Professor Claus



WS 2002/2003
Bertele, Lewandowski
15. November 2002

Aufgabenblatt 5

Abgabe bis 21.11.2002 20:00 Uhr – Besprechung in der Woche ab dem 25.11.2002

Aufgabe 1 Klausuraufgabe H2002/I-1/1 (schriftlich, leicht) 1+¼+¼+½ Punkte

So jetzt kommt die erste Klausuraufgabe zum eigenen Training. Bei Grammatiken, Sprachen und Syntaxdiagrammen ist es wichtig auch auf die formale Korrektheit zu achten. Werden beispielsweise nur die Produktionsregeln angegeben ist das wie ein Programm ohne Deklaration der Variablen. Prinzipiell folgen zwar aus den Produktionsregeln, die anderen Mengen und das Tupel, aber im Bereich von Spezifikationen, egal ob Daten (als Sprache) oder Algorithmen (als Programme), ist etwas mehr Formalismus nicht schlecht.

a) Geben Sie eine kontextfreie Grammatik G mit höchstens fünf Produktionsregeln an, die eine endliche Sprache mit mehr als 10 Wörtern beschreibt.
b) Geben Sie 5 Wörter aus dieser Sprache an.
c) Geben Sie die von Ihnen beschriebene Sprache in Mengenschreibweise an.
d) Geben Sie die von Ihnen beschrieben Sprache als Syntaxdiagramm an.

Aufgabe 2 Klausuraufgabe H2001/I-1/1 (Votieraufgabe, leicht) ½+½+½+½ Punkte

Und die zweite Klausuraufgabe – gestellt ein Jahr früher.

Gegeben sind folgende Grammatiken:
$G_1 = (N_1,T,P_1,S_1)$ mit $N_1=\{S_1\}$, $T=\{a,b\}$, $P_1=\{S_1 \rightarrow aS_1 | bS_1 | \varepsilon\}$ und $G_2=(N_2,T,P_2,S_2)$ mit $N_2=\{S_2\}$, $P_2=\{ S_2 \rightarrow S_2\, a | S_2 b | \varepsilon\}$, sowie $G_3=(N_3,T,P_3,S_3)$ mit $N_3=\{S_3\}$, $P_3=\{S_3 \rightarrow aS_3 | a\}$.

a) Geben Sie $L(G_1)$ in Mengenschreibweise an.
b) Geben Sie $L(G_1) \cup L(G_2)$ in Mengenschreibweise an.
c) Geben Sie $L(G_1) \cap L(G_2)$ in Mengenschreibweise an.
d) Geben Sie $L(G_1) \cap L(G_3)$ in Mengenschreibweise an.

Aufgabe 3 Klausuraufgabe H2001/I-1/5 (Votieraufgabe, leicht) 2 Punkte

Manchmal ist das komplizierte an einer Aufgabe eine Einschränkung der Mittel. Die folgende Aufgabe zeigt, dass auch eine etwas aufwändigere, kontextfreie Sprache mit nur einem Syntaxdiagramm darstellbar ist.

Zeichnen Sie ein einziges Syntaxdiagramm (das auch Schleifen enthalten darf), das die Sprache $L=\{a^n b^m \mid n \geq m \geq 1\}$ erzeugt. Die Verwendung mehrerer Diagramme ist (hier) unzulässig!

Aufgabe 4 Asymptotische Komplexitätsbeziehungen (Votier., mittel) 1,5+0,5+2 Punkte

Diese Aufgabe soll Ihnen das Verständnis der Komplexitätsklassen vermitteln.

a) Zeigen oder widerlegen Sie folgende Aussagen:
- $3\log_{10} n \in O(\log_2 n)$
- $(n+a)^b \in O(n^b),\ a,b \in \mathbb{R}$ und $b>0$
- $3^n \in O(2^n)$

b) Zeigen Sie anhand der Definitionen von *O* und *Ω*, dass gilt:
$f(n) \in O(g(n)) \Leftrightarrow g(n) \in \Omega(f(n))$

c) Welche Beziehung $(=, \subseteq, \supseteq)$ gilt zwischen den folgenden Komplexitäten:
- $O(n\sqrt{n})$ und $O(n\log^2(n))$
- $O(3^n)$ und $O(n^n)$
- $O(2^{n+1})$ und $O(2^n)$
- $O(2^{2n})$ und $O(2^n)$

Aufgabe 5 *O*-Notation (Votieraufgabe, mittel) 1 Punkt

Und jetzt mal so etwas...

Finden Sie einen Fehler im folgenden Argument.
Behauptung: $f(n) = 2^n \in O(1)$
Beweis: Vollständige Induktion nach *n*. Die Behauptung ist richtig für $n = 1$, denn $2^1 = 2 \in O(1)$. Die Induktionsannahme ist nun also, dass $2^{n-1} \in O(1)$. Zu zeigen ist demnach, das dann auch $2^n \in O(1)$. Das ist einfach, weil $2^n = 2 \cdot 2^{n-1} = 2 \cdot O(1) = O(1)$.

Aufgabe 6 Analyse von Algorithmen (schriftlich, mittel) 1+1+2 Punkte

Oft bekommt man als Informatiker einen Algorithmus gegeben und muss beurteilen, ob dieser brauchbar ist.

Gegeben sei folgender Algorithmus, der auf dem Feld *A(1..n)* arbeitet:

```
for i:= 1 to n-1 loop
  for j:= i+1 to n loop
    if A(i) > A(j) then "vertausche A(i) und A(j)" end if;
  end loop;
end loop;
```

a) Was macht dieser Algorithmus?
b) Wie groß ist seine Komplexität in der O-Notation?
c) Wie lautet die exakte Komplexität? (Berücksichtigen Sie dabei nur die Anzahl der Vergleiche.)

Aufgabe 7 Das Totalitätsproblem (schriftlich, schwer) 5 Punkte

Das Zurückführen eines Problems auf ein anderes, ist in der theoretischen Informatik sehr wichtig. In der praktischen Informatik brauchen wir dieses Wissen um uns Arbeit zu ersparen. Wenn wir zeigen können, dass das zu lösendes Problem ein „unlösbares Problem" enthält, brauchen wir gar nicht erst mit der Programmierung beginnen.

Das Totalitätsproblem ist wie folgt definiert:
„Geben Sie einen Algorithmus an, der zu jedem Programm π feststellt, ob die von π realisierte Abbildung $f_\pi: E_\pi \rightarrow A_\pi$ eine totale Funktion ist.
(Beziehungsweise: Geben Sie einen Algorithmus an, der für jedes Programm π feststellt, ob π immer terminiert, vergleiche 1.1.3.)"
Zeigen Sie durch Rückführung auf das Halteproblem: Das Totalitätsproblem ist algorithmisch nicht lösbar.

Aufgabe 8 Modifizierung der Charakteristika von Algorithmen (Zusatzaufgabe, mittel) 4 Punkte

Wenn wir doch nur bessere Mittel zur Verfügung hätten, wäre die Welt doch gleich ganz anders...

Überlegen Sie sich, was sich ändert, wenn man in der Definition des Begriffs Algorithmus die Forderung weglässt, dass die Berechnungsvorschrift endlich sein muss (Hinweis: Entwerfen Sie einen Algorithmus mit unendlicher Berechnungsvorschrift, der das Halteproblem löst). Beachten Sie, dass die Laufzeit Ihres Algorithmus nichtsdestotrotz stets endlich sein muss, es wird nur ein Großteil der Berechnungsvorschrift nicht ausgeführt.

Allgemeine Hinweise:

- Hinweis zu Ada – Für alle, die etwas gegen oder für Ada haben:
  http://www.seas.gwu.edu/~mfeldman/ada-project-summary.html
  http://lglwww.epfl.ch/Ada/
  http://www.seas.gwu.edu/~adagroup/adalib_html/ada-html/index.html
  http://www.cl.cam.ac.uk/~mgk25/ada.html
  http://www.adapower.com/reuse/index.html
  http://www.adaic.com/whyada/WhyAda.html
- Bitte beachten Sie, dass die Tutoren nur reine Textdateien (*.TXT, *.HTM[L]) akzeptieren. Binärdateien und andere Dateiformate bitte nur nach Absprache mit dem jeweiligen Tutor.
- Es sind auf diesem Aufgabenblatt 24 Punkte erreichbar. Davon werden für den Übungsschein maximal 20 Punkte angerechnet. Die Zusatzaufgabe wird eventuell in den Übungen nicht besprochen. Die Bearbeitung ist nicht verpflichtend, dient jedoch zum eigenen Training.
- Hinter jeder Aufgabe ist die Art der Aufgabe, der Schwierigkeitsgrad, sowie die erreichbare Punktzahl angegeben.
- Die schriftlichen Aufgaben (11 Punkte) geben Sie bitte zum Abgabezeitpunkt im eClaus-System ab. Bitte votieren Sie bitte ebenfalls bis zum Abgabezeitpunkt im eClaus-System.
- Ada-Umgebung:
  In der EfidI 1/2, sowie den begleitenden Veranstaltungen verwenden wir Ada95 als Programmiersprache. An unserem Institut wird hierzu die GNAT-Umgebung verwendet, die frei zu Verfügung gestellt wird. Im Grundstudiumspool ist GNAT sowohl unter Windows, als auch unter Linux installiert. Für Windows gibt es ferner die kostenlose Entwicklungsumgebung GIDE, welche auch im GS-Pool installiert ist. Falls Sie Ada nicht am Institut verwenden wollen und keine Downloadmöglichkeit (http://www.lern-plus.de/stuetzkurs/defaultSS02.htm) haben, so können Sie donnerstags nach der Vorlesung bei Stefan Lewandowski kostenlos eine CD mit allen wichtigen Werkzeugen erhalten. (Natürlich nur solange der Vorrat reicht.) Eine Anleitung zu GNAT unter Windows befindet sich im Anhang dieses Blatts.
- Bei weiteren Fragen, wenden Sie sich bitte an das Forum (http://fachschaft.informatik.uni-stuttgart.de/forum/), Ihren Tutor, oder per Mail direkt an J. Bertele (inf@studbs.informatik.uni-stuttgart.de).

Weitere Hinweise finden Sie auf unserer Veranstaltungswebseite unter:
http://www.informatik.uni-stuttgart.de/ifi/fk/lehre/ws02-03/info_I_0203.html