Prof. Dr. Hans-Jörg Kreowski, Dr. Sabine Kuske
Studiengang Informatik
Linzer Str. 9a
OAS 3001, 3005
Tel.: 2956, 2335, 3697 (Sekr.), Fax: 4322
E-Mail: {kreo,kuske}@informatik.uni-bremen.de
www.informatik.uni-bremen.de/theorie

Juli 2004
Fragenkatalog

Theoretische Informatik 2

Ein paar Fragen

Wie verabredet, soll am Ende der Lehrveranstaltung ein kurzes Fachgespräch stattfinden, in dem von jeder Teilnehmerin und jedem Teilnehmer gemäß Prüfungsordnung einige Fragen beantwortet werden sollen. Zu einem Fachgespräch könnt ihr einzeln oder in Gruppen bis zu vier Personen erscheinen, wobei bei einem Gruppenfachgespräch natürlich jeder einzeln befragt wird. Die folgenden Fragenkataloge sind dafür die Grundlage. Dabei geht es nicht nur darum, eine einzelne Frage beantworten zu können, sondern über die thematischen Komplexe (wie Wörter, CE-S, Sortieren, Matrizenmultiplikation, monotone Grammatiken, kontextfreie Grammatiken, Turing-Maschinen, *while-S* usw.) etwas zu wissen und sagen zu können. Außerdem kommt es nicht nur auf die Konzepte an (Katalog 1), sondern auch auf deren Eigenschaften (Katalog 2).

Katalog 1: Was ist das? Wie ist es konstruiert?

1. Linksaddition
2. Konkatenation von Wörtern
3. Menge aller Wörter über einem Alphabet
4. Konkatenation von Sprachen
5. Induktionsprinzip für Wörter
6. CE-S-Spezifikation
7. Term
8. Werteterm
9. Bedingte Gleichung
10. Wertzuweisung
11. Substitution
12. Auswertungsschritt
13. Gleichungsanwendung
14. Berechnung einer CE-S-Spezifikation
15. Gleichwertigkeit von Termen
16. Vorwärtsinterpreter für CE-S
17. Gleichungsbeweiser für CE-S
18. CE-S-berechenbare Funktion
19. Aufwand einer CE-S-Spezifikation
20. das große O
21. Aufwandsklasse
22. die Klasse P
23. die Klasse NP
24. NP-Vollständigkeit
25. P=NP-Problem
26. die Klasse $PSPACE$
27. die Klasse $NPSPACE$
28. Chomsky-Grammatik
29. Produktion
30. Anwendung einer Produktion
31. direkte Ableitung
32. Ableitung
33. von einer Chomsky-Grammatik erzeugte Sprache
34. Typ-3-Grammatik
35. Typ-2-Grammatik
36. Typ-1-Grammatik
37. Typ-0-Grammatik
38. monotone Grammatik
39. kontext-sensitive Grammatik
40. kontextfreie Grammatik

41. rechtslineare Grammatik
42. reguläre Grammatik
43. Wortproblem
44. Entscheidbarkeit eines Problems
45. Postsches Korrespondenzproblem
46. Cocke-Kasami-Younger-Verfahren
47. erweiterter endlicher Automat
48. Turing-Maschine
49. Konfiguration
50. deterministische Turing-Maschine
51. Turing-berechenbare Funktion
52. *while*-S-Programm
53. graphische Darstellung eines *while*-S-Programms
54. Berechnung eines *while*-S-Programms
55. Churchsche These

Katalog 2: Was gilt? Was nicht? Warum?

1. Die Konkatenation von Wörtern ist assoziativ.
2. Die Konkatenation von Wörtern ist kommutativ.
3. Das Sortierproblem liegt in $O(n)$.
4. Das Sortierproblem liegt in $O(n \cdot \mathrm{ld}\, n)$.
5. Das Sortierproblem liegt in $O(n^2)$.
6. Der klassische Algorithmus für Matrizenmultiplikation benötigt n^3 Multiplikationen und $n^2 * (n-1)$ Additionen.
7. $O(ldn) \subseteq O(n) \subseteq O(n^2) \subseteq O(n^3) \subseteq \cdots \subseteq O(2^n)$.
8. $P \subseteq NP$.
9. $NP \subseteq P$.
10. $P \subseteq PSPACE$.
11. $PSPACE \subseteq P$.
12. $PSPACE \subseteq NPSPACE$.
13. $NPSPACE \subseteq PSPACE$.
14. $NP \subseteq NPSPACE$.
15. $NPSPACE \subseteq NP$
16. Das Wortproblem für monotone Grammatiken ist in *NPSPACE*.
17. Das Wortproblem für kontextfreie Grammatiken ist in kubischer Zeit lösbar.
18. Das Wortproblem für Chomsky-Grammatiken ist entscheidbar.
19. Das Wortproblem ist für monotone Sprachen entscheidbar.
20. Das Wortproblem ist für kontextfreie Sprachen nicht entscheidbar.
21. Das Leerheitsproblem ist für kontexfreie Sprachen entscheidbar.
22. Die Leerheit des Durchschnitts zweier kontextfreier Sprachen ist entscheidbar.
23. Die Leerheit des Durchschnitts zweier regulärer Sprachen ist entscheidbar.

24. Das Postsche Korrespondenzproblem ist entscheidbar.
25. Jede Turing-berechenbare Funktion ist CE-S-berechenbar und umgekehrt.
26. Jede *while*-S-berechenbare Funktion ist CE-S-berechenbar und umgekehrt.
27. Jede Turing-berechenbare Funktion ist *while*-S-berechenbar und umgekehrt.