Lehrstuhl für Informatik 1
Prof. Dr. Berthold Vöcking & Prof. Dr. Martin Grohe
Kamal Al-Bawani
Benjamin Ries

WS 2013/14
20.12.2013

Übung zur Vorlesung
Berechenbarkeit und Komplexität

Fragenkatalog

Die Klausur und die Präsenzübungen bestehen zu einem großen Teil aus Wissens- und einfachen Verständnisfragen. Der nachfolgende Fragenkatalog soll dazu dienen, das eigene Wissen zu kontrollieren und ein Gefühl für mögliche Aufgabenstellungen zu bekommen.
Einige dieser Fragen werden sich sicherlich auch in den Klausuren wiederfinden.

Fragen zur Einführung:

1. Sei $\Sigma = \{0, 1\}$. Wofür steht $\Sigma^k, \Sigma^0, \Sigma^*$? Was ist eine *Sprache* über dem Alphabet Σ?
2. Definieren Sie das Modell der Turingmaschine.
3. Wann ist eine Sprache *rekursiv (entscheidbar)*?
4. Beschreiben Sie formal eine TM, die die Sprache $\{0^i 1^j \mid i, j \geq 0\}$ erkennt. Geben Sie insbesondere die Übergangsfunktion an.
5. Was ist die Konfiguration einer TM? Was ist eine Nachfolgekonfiguration?
6. Was ist der Unterschied zwischen einer Mehrspur- und einer Mehrband-TM?
7. Beschreiben Sie die Simulation einer k-Band TM durch eine 1-Band TM und geben Sie die Laufzeit der Simulation an.
8. Was ist eine Gödelnummerierung?
9. Was versteht man unter der universellen Turingmaschine?
10. Beschreiben Sie die Implementierung der universellen Turingmaschine auf einer 1-Band TM und geben Sie den Zeitverlust an.
11. Definieren Sie das Modell der RAM.
12. Was ist der Unterschied zwischen dem uniformen und dem logarithmischen Kostenmaß der RAM?
13. Beschreiben Sie die Simulation der RAM durch eine k-Band-TM und geben Sie die Laufzeit der Simulation an.
14. Beschreiben Sie die Simulation einer 1-Band TM durch eine RAM und geben Sie die Laufzeit der Simulation an.
15. Was besagt die Church-Turing-These?

Fragen zur Berechenbarkeit:

1. Warum existieren nicht-rekursive Sprachen?
2. Definieren Sie die folgenden Entscheidungsprobleme und ihre Komplemente in Form einer Sprache. Wie kann man zeigen, dass diese Probleme nicht rekursiv sind?
   a) Diagonalsprache

   $$D = \{ \ldots\ldots \mid \ldots\ldots \} \qquad \bar{D} = \{ \ldots\ldots \mid \ldots\ldots \}$$

   Halteproblem

   $$H = \{ \ldots\ldots \mid \ldots\ldots \} \qquad \bar{H} = \{ \ldots\ldots \mid \ldots\ldots \}$$

   Spezielles Halteproblem

   $$H_\epsilon = \{ \ldots\ldots \mid \ldots\ldots \} \qquad \bar{H}_\epsilon = \{ \ldots\ldots \mid \ldots\ldots \}$$

   Allgemeines Halteproblem

   $$H_{\text{all}} = \{ \ldots\ldots \mid \ldots\ldots \} \qquad \bar{H}_{\text{all}} = \{ \ldots\ldots \mid \ldots\ldots \}$$

3. Was besagt der Satz von Rice?
4. Wie kann man mit Hilfe des Satzes von Rice zeigen, dass ein Probleme nicht rekursiv ist?
5. Wann *entscheidet* und wann *erkennt* eine TM eine Sprache?
6. Wann ist eine Sprache *rekursiv aufzählbar (semi-entscheidbar)*?
7. Sind rekursive bzw. rekursiv aufzählbare Sprachen abgeschlossen gegen
   a) Schnitt
   b) Vereinigung
   c) Komplement?

   Begründen Sie Ihre Antworten.
8. Gegeben sei eine TM M, die L erkennt und eine TM $\bar{M}$, die $\bar{L}$ erkennt. Ist dann L entscheidbar? Begründen Sie Ihre Antwort.
9. Nennen Sie jeweils ein Beispiel für eine nicht-rekursive Sprache L, so dass gilt
   (a) L rekursiv aufzählbar und $\bar{L}$ nicht rekursiv aufzählbar,
   (b) L nicht rekursiv aufzählbar und $\bar{L}$ rekursiv aufzählbar,
   (a) L und $\bar{L}$ nicht rekursiv aufzählbar.
10. Wie lautet die Definition für die *Reduktion* ($L_1 \leq L_2$)?
11. Was unterscheidet die Reduktion von der Unterprogrammtechnik?
12. Wiederholen Sie die Übungsaufgaben zur
    (a) Diagonalisierung,
    (b) Unterprogrammtechnik,
    (c) Reduktion.

    Hinweis: Gliedern Sie Ihre Beweise analog zu den in der Vorlesung vorgestellten Beweisen.

13. Definieren Sie das zehnte Hilbertsche Problem und das Post'sche Korrespondenzproblem.
14. Warum gilt $H \leq PKP$? Wie kann eine TM durch Dominos simuliert werden?
15. Wann ist ein Rechnermodell Turing-mächtig?
16. Definieren Sie die Programmiersprache WHILE.
17. Warum ist die Programmiersprache WHILE Turing-mächtig?
18. Definieren Sie die Programmiersprache LOOP.
19. Wann ist eine Funktion primitiv-rekursiv?
20. Nennen Sie eine nicht primitiv-rekursive Funktion.
21. Warum ist die Programmiersprache LOOP nicht Turing-mächtig?

Fragen zur Komplexität:

1. Wie ist die Worst-Case Laufzeit eines Algorithmus definiert?
2. Was ist ein Polynomialzeitalgorithmus?
3. Definieren Sie die Komplexitätsklasse P.
4. Was ist eine nicht-deterministische Turingmaschine (NTM)? Wie sind ihr Akzeptanzverhalten und ihre Laufzeit definiert?
5. Geben Sie die Simulation einer NTM mit polynomieller Worst-Case Laufzeit durch eine deterministische TM an. Wie hoch ist die Laufzeit der resultierenden TM?
6. Erläutern und beweisen Sie, in welcher Beziehung die Komplexitätsklassen P, NP und EXPTIME zueinander stehen.
7. Geben Sie die Optimierungs- und Entscheidungsvarianten der folgenden Probleme an: Clique, KP, BPP, TSP.
8. Zeigen Sie für die obigen Probleme, dass sich aus polynomiellen Algorithmen für die Entscheidungsvarianten polynomielle Algorithmen für die Optimierungsvarianten folgern lassen, und umgekehrt.
9. Definieren Sie die Komplexitätsklasse NP mit Hilfe von NTMs einerseits und Polynomialzeitverifizierern andererseits. Zeigen Sie die Äquivalenz der beiden Definitionen.
10. Nennen Sie den Unterschied zwischen $\leq$ und $\leq_p$.
11. Wann heißt ein Problem NP-hart bzw. NP-vollständig?
12. Beschreiben Sie die Aussage und die Beweisidee des Satzes von Cook und Levin. (Wiederholen Sie dazu sorgfältig den Beweis.)
13. Wiederholen Sie sämtliche polynomielle Reduktionen aus der Vorlesung und den Übungen.