Prof. Dr. Heinz Schmitz
Fachbereich Informatik
Hochschule Trier

Berechenbarkeit und Komplexität
Wintersemester 2018/2019
21. September 2018

Inhalt

Teil I – Berechenbarkeit

1. Einführung

   1.1. Über diese Lehrveranstaltung
   Themenübersicht, Lernziele Wissen / Anwenden / Beweisen, Voraussetzungen, Software, Tipps.

   1.2. WHILE-Programme (Wdh.)
   Berechnungsmodelle, Syntax und Semantik von WHILE-Programmen, die von einem WHILE-Programm P berechnet Funktion $\varphi_P : \mathbb{Z}^m \mapsto \mathbb{Z}$, Nichtdefiniertheit, Beispiele `prodZ`, `divZ`, Bestimmung der berechneten Funktion.

   1.3. Codierungen
   Codierung $code_Z$ zwischen $\mathbb{Z}$ und $\mathbb{N}$, $f_{\mathbb{N}}$, dyadische Darstellung dya, Berechnungsvorschrift für dya und dya^{-1}, k-adische Darstellung ad_k, Berechnungsvorschrift für ad_k und ad_k^{-1}, Codierung $code_{\Sigma^*}$ zwischen $\mathbb{N}$ und Σ^*, $f_{\mathbb{N}}$, Listencodierung $\langle x_0, \ldots, x_{m-1} \rangle$, Grundbereiche $\mathbb{G}$.

2. Turingmaschinen

   2.1. Berechnungsmodell
   Aufbau und Arbeitsweise, Überführungsfunktion, Beispiel, TM-Konfiguration, Ausführung eines TM-Befehls, Definition Turingmaschine $M = (\Sigma, \Gamma, Z, f, z_0, z_1)$, Konventionen, TM-Standardrepräsentation in PYTHON, TM `inc`.

   2.2. Turing-Berechenbarkeit
   TM-Startkonfiguration $(x_0 * \cdots * x_{m-1}, z_0, 0)$, TM-Berechnungsergebnis bei Stopp, berechnete Funktion $\varphi_M : (\Sigma^*)^m \mapsto \Sigma^*$, Turing-Berechenbarkeit von Funktionen auf Σ^*; TM M hält genau dann an, wenn $\varphi_M(x)$ definiert ist; Beispiel, Berechnung von $\varphi_M(w)$ mit `run_tm`.

3. Berechenbare Funktionen

   3.1. WHILE- versus Turing-Berechenbarkeit
   WHILE-Berechenbarkeit für Funktionen auf $\mathbb{Z}$, die Klassen **WHILE**, **TM** und $\mathbf{WHILE}_{\mathbb{N}}$, $\mathbf{TM}_{\mathbb{N}}$, $f \in \mathbf{WHILE}_{\mathbb{N}}$ gdw. Fkt. $g \in \mathbf{WHILE}$ ex. die mit f auf $\mathbb{N}$ übereinstimmt; $f \in \mathbf{TM}_{\mathbb{N}}$ gdw. Fkt. $g \in \mathbf{TM}$ ex. die f in dyadische Darstellung berechnet; Satz: $\mathbf{TM}_{\mathbb{N}} \subseteq \mathbf{WHILE}_{\mathbb{N}}$, Simulation von TMs durch WHILE-Programme; Satz: $\mathbf{WHILE}_{\mathbb{N}} \subseteq \mathbf{TM}_{\mathbb{N}}$ o.B., Kleenesches Normalformtheorem.

   3.2. Algorithmusbegriff
   Hauptsatz $\mathbf{WHILE}_{\mathbb{N}} = \mathbf{TM}_{\mathbb{N}} = ..$, Turing-Vollständigkeit, Church-Turing-These, Definition Algorithmus und Berechenbarkeit, Abschluss unter Hintereinanderausführung.

   3.3. **Exkurs:** Historische Notizen
   Frage der Axiomatisierbarkeit der Mathematik, Geschichte des Algorithmenbegriffs.

4. Eigenschaften berechenbarer Funktionen

   4.1. Gödelisierung
   Codierung $code(M)$ von Turingmaschinen, Gödelisierung, Gödelnummer i, Turingmaschinen M_i mit Nummer, i-te m-stellige berechenbare Funktion $\varphi_i^{(m)}$, Notationen $\varphi_i(x)\downarrow$, $\varphi_i(x)\uparrow$, $M_i(x)\downarrow$, $\mathcal{M}_i(x)\uparrow$.

   4.2. Universelle Algorithmen
   Aufzählungssatz, Existenz universeller Algorithmen, Beispiel PYTHON-Shell, Paddinglemma, s-m-n-Theorem, Konstruktion neuer Programme.

   4.3. **Exkurs:** Rekursionssatz von Kleene
   Rekursionssatz, Selbstreproduktionssatz.

5. Entscheidbarkeit und Aufzählbarkeit

   5.1. (Semi-) Entscheidbare Mengen
   Entscheidungsproblem, charakteristische Funktion c_A, semi-charakteristische Funktion χ_A, Beispiele Goldbachsche Vermutung und Collatz-Problem, (Semi-) Entscheidbarkeit, Klasse **REC**, Abschlußeigenschaften, Satz: A ist entscheidbar gdw. A und $\overline{A}$ semi-entscheidbar sind.

   5.2. Aufzählbare Mengen
   Aufzählende Funktion $f : \mathbb{N} \mapsto \mathbb{N}^m$ mit $W_f = A$, Aufzählbarkeit, Klasse **RE**, Cantorsche Paarungsfunktion π, Beispiel Primzahldifferenzen, jede aufzählbare Menge ist semi-entscheidbar.

5.3. Eigenschaften entscheidbarer und aufzählbarer Mengen
Mehrstellige aufzählende Funktionen $f : \mathbb{N}^k \mapsto \mathbb{N}^m$, Satz: Charakterisierungen der aufzählbaren Mengen; Abschlusseigenschaften, Projektion, Satz: A ist aufzählbar gdw. A Projektion einer entscheidbaren Menge; Zusammenfassung zum Nachweis von Entscheidbarkeit und Aufzählbarkeit.

6. Das Halteproblem

6.1. Unentscheidbarkeit
Halteproblem K, spezielles Halteproblem K_0, Satz: K_0 ist aufzählbar, aber nicht entscheidbar; $\overline{K_0}$ ist nicht aufzählbar, **RE** ist nicht unter Komplement abgeschlossen, K ist aufzählbar, aber nicht entscheidbar.

6.2. Many-one-Reduzierbarkeit
Definition, $\leq_m$ ist Quasiordnung, **RE** und **REC** sind unter $\leq_m$ abgeschlossen, Satz: $K_0 \equiv_m K$.

6.3. Many-one-Vollständigkeit
Die nichttrivialen entscheidbaren Mengen sind $\leq_m$-äquivalent, Beispiel, RE-Vollständigkeit, Satz: K ist RE-vollständig; **Exkurs:** Struktur der m-Grade, Satz: Es gibt unentscheidbare und bzgl. $\leq_m$ unvergleichbare Mengen.

7. Unentscheidbare Mengen

7.1. Satz von Rice
TOTAL, Satz: TOTAL ist unentscheidbar (durch Reduktion von K_0); Indexmenge, Satz: Auf jede nichttriviale Indexmenge lässt sich K_0 oder $\overline{K_0}$ reduzieren; alle nichttrivialen Indexmengen sind unentscheidbar, enthalten sie den Index der nirgends definierten Funktion, dann noch nicht einmal aufzählbar; Satz von Rice: Ist $\mathcal{C} \neq \emptyset$ eine echte Teilmenge aller berechenbaren Funktionen, dann ist $I(\mathcal{C})$ unentscheidbar.

7.2. Übersicht Teil I – Berechenbarkeit
Übersichtsabbildungen.

Teil II – Komplexität

8. Grundlagen zur Komplexität

   8.1. Laufzeitanalyse (Wdh.)
   Vorgehensweise, Schrittzahlfunktion STEP_M, Eingabelänge, Laufzeitfunktion TIME_M, PYTHON-Programme, vereinfachte Laufzeitanalyse.

   8.2. Polynomial- und Exponentialzeit (Wdh.)
   Definition Polynomial- und Exponentialzeit, Klassen $\mathbf{P}$, $\mathbf{FP}$, $\mathbf{EXP}$, Diskussion zur praktischen Machbarkeit.

   8.3. Berechnungsprobleme
   Katalog der Entscheidungsprobleme KS, SOS, BP, PART, SAT, CNF-SAT, 3-CNF-SAT, TSP, HC, CLIQUE, VC, PUZZLE, gemeinsame Problemstruktur.

9. P-NP-Theorie

   9.1. Die Klasse $\mathbf{NP}$ (Wdh.)
   Motivation und Definition $\mathbf{NP}$, SOS, KNAPSACK und TSP sind in $\mathbf{NP}$, Verifikationsalgorithmus `V_sos`; Satz: $\mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{EXP}$, `sos_exhaustive`, P-NP-Frage; weitere Beispielmengen in $\mathbf{NP}$, polynomielle Projektionen, Vergleich mit **RE**.

   9.2. NP-Vollständigkeit
   Polynomialzeit-Many-one-Reduzierbarkeit (Wdh.): $\leq_m^p$ mit $f \in \mathbf{FP}$, SOS $\leq_m^p$ KS, Quasiordnung, $\leq_m^p$-Abschluss von $\mathbf{P}$ und $\mathbf{NP}$; NP-Vollständigkeit, Eigenschaften NP-vollständiger Probleme.

   9.3. Vollständigkeitsbeweis
   Satz: PUZZLE ist NP-vollständig. Beweis über Simulation von Polzeit-TM-Berechnungen durch PUZZLE-Instanzen.

10. Reduktionen

   10.1. Vorgehensweise
   Übersicht; Spezialisierung, lokale Ersetzung, Komponenten-Design.

   10.2. Reduktionen von 3-CNF-SAT
   Idee / Beispiel / Konstruktion / Korrektheit, Satz: 3-CNF-SAT $\leq_m^p$ SOS; Satz: 3-CNF-SAT $\leq_m^p$ VC; Definition DHC, Satz: 3-CNF-SAT $\leq_m^p$ DHC; Folgerung: SOS, VC und DHC sind NP-vollständig.

11. Streng NP-vollständige Probleme

11.1. Strenge NP-Vollständigkeit
Beispiel PART-Instanzen, $max_{\mathcal{P}}(x)$, q-Subproblem $\mathcal{P}_q$, *strenge* NP-Vollständigkeit, SAT, CNF-SAT, 3-CNF-SAT, HC, CLIQUE, VC, PUZZLE sind streng NP-vollständig, *Large Number Problem* (LNP), KS, SOS, BP, PART, TSP sind LNPs; falls $\mathcal{P}$ kein LNP, ist $\mathcal{P}$ NP-vollständig gdw. $\mathcal{P}$ streng NP-vollständig.

11.2. Pseudo-Polynomialzeit
Beispiel PART, Berechnung aller $T[i, s]$ mittels Dynamischer Programmierung, `part_dp`, PART ist in $O(mS)$ entscheidbar, Pseudo-Polynomialzeit $O(p(|x|, max_{\mathcal{P}}(x)))$; hat $\mathcal{P}$ einen Pseudo-Polzeitalgorithmus, dann sind alle $\mathcal{P}_q$ in Polzeit lösbar; Folgerung: PART_q, KS_q, $\text{SOS}_q \in \mathbf{P}$; hat ein streng NP-vollständiges Problem einen Pseudo-Polzeitalgorithmus, folgt $\mathbf{P} = \mathbf{NP}$.

11.3. Streng NP-vollständige LNPs
TSP ist streng NP-vollständig wegen HC $\leq_m^p$ TSP_q; TRIPLE PARTITION ist streng NP-vollständig (o.B.); BP ist streng NP-vollständig wegen pseudo-polynomieller Reduktion TPART $\leq_m^p$ BP; Methoden zum Nachweis strenger NP-Vollständigkeit, P/NP-Übersichtsabbildungen.

12. NP-Optimierungsprobleme

12.1. Modellierung
Definition Optimierungsproblem, $\mathcal{I}$, sol, m, $goal$, y^*, sol^*, m^*, $\mathcal{P}_C$, $\mathcal{P}_E$, $\mathcal{P}_D$; Beispiele MAX KS, MIN BP, MIN TSP, MAX CLIQUE, MIN VC, die Klasse $\mathbf{NPO}$, Beispiel MIN VC, Problemstruktur; Satz: Jedes $\mathcal{P} \in \mathbf{NPO}$ lässt sich in Zeit $O(2^{p(n)})$ lösen; Satz: Für jedes $\mathcal{P} \in \mathbf{NPO}$ ist $\mathcal{P}_D \in \mathbf{NP}; \mathbf{PO}$.

12.2. Turing-Reduzierbarkeit und NP-harte Probleme
Orakel, Definition Turing-Reduzierbarkeit, $\leq_T^p$ ist reflexiv und transitiv, aus $\leq_m^p$ folgt $\leq_T^p$; $\mathcal{P}_D \leq_T^p \mathcal{P}_E \leq_T^p \mathcal{P}_C$; NP-Härte; ist ein NP-hartes Problem effizient lösbar, folgt $\mathbf{P} = \mathbf{NP}$; ist $\mathcal{P}_D$ NP-hart, dann auch $\mathcal{P}$; MAX KS, MIN BP, MIN TSP, MAX CLIQUE, MIN VC sind NP-hart, strenge NP-Härte, MIN BP, MIN TSP, MAX CLIQUE, MIN VC sind streng NP-hart.

12.3. Die PO-NPO-Frage
Für alle $\mathcal{P} \in \mathbf{NPO}$ gilt $\mathcal{P}_E \leq_T^p \mathcal{P}_D$ und es ex. ein $\mathcal{P}' \in \mathbf{NPO}$ mit $\mathcal{P}_C \leq_T^p \mathcal{P}'_E$; Satz: $\mathbf{P} = \mathbf{NP} \Leftrightarrow \mathbf{PO} = \mathbf{NPO}$.

13. Approximationsalgorithmen
Lösungsansätz für NP-harte Optimierungsprobleme.

13.1. Performanzrate
Definition, Beispiel, r-Approximationsalgorithmus.

13.2. Beispiele mit konstanter Performanzrate
Greedy-Algorithmus `max_ks_greedy` für MAX KS ohne Gütegarantie, Satz: Ergänzung v_{max} liefert 2-Approximation; Satz: `min_bp_next_fit` ist 2-Approximation für MIN BP; Online-Eigenschaft, Verbesserung durch `min_bp_first_fit`, Satz: `min_bp_first_fit_decreasing` garantiert $m(x, y) \leq \frac{11}{9} m^*(x) + 1$.

13.3. Approximierbarkeit
Die Klasse $\mathbf{APX}$, MAX KS, MIN BP $\in \mathbf{APX}$, Satz: Falls MIN TSP $\in \mathbf{APX}$, dann ist $\mathbf{P} = \mathbf{NP}$; Satz: Existiert für MIN BP ein r-Approximationsalgorithmus mit $r < \frac{3}{2}$, dann ist $\mathbf{P} = \mathbf{NP}$; Approximationsschema, Übersicht.

A Grundbegriffe

Index

Literatur

[1] O. Goldreich. *P, NP, and NP-Completeness – The Basics of Computational Complexity*. Cambridge University Press, 2010.

[2] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

[3] K. W. Wagner. *Theoretische Informatik – Eine kompakte Einführung*. Springer, second edition, 2003.

[4] J. Hromkovič. *Theoretische Informatik – Formale Sprachen, Berechenbarkeit, Komplexitätstheorie, Algorithmik, Kommunikation und Kryptographie*. Teubner, third edition, 2007.

[5] G. Vossen and K.-U. Witt. *Grundkurs Theoretische Informatik*. Vieweg, third edition, 2004.

[6] H. Rogers Jr. *Theory of Recursive Functions and Effective Computability*. MIT–Press, 1967. third printing 1992.

[7] I. Wegener. *Theoretische Informatik – Eine algortihmenorientierte Einführung*. Teubner, 1993.

[8] U. Schöning. *Theoretische Informatik kurzgefaßt*. Spektrum, 2001.

[9] C. H. Papadimitriou. *Computational Complexity*. Addison–Wesley, 1994.

[10] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi. *Complexity and Approximation – Combinatorial Optimization Problems and Their Approximability Properties*. Teubner, 2003. second corrected printing.