Theoretische Grundlagen der Informatik

Tutorium 7 — Wintersemester 2014/15

Moritz Klammler — 25. November 2014

Tagesthemen

- Many-to-One-Reduzierbarkeit
- Satz von Rice
- Kodierungsschemata
- Die Sprachklassen $\mathcal{P}$ und $\mathcal{NP}$

Many-to-One-Reduzierbarkeit

Sei Σ ein Alphabet und $B \subseteq \Sigma^*$ (semi-)entscheidbar, sowie $A \subseteq \Sigma^*$.

Wenn es eine *berechenbare* Funktion $f : \Sigma^* \to \Sigma^*$ gibt, sodass für alle $w \in \Sigma^*$ gilt

$$w \in A \Leftrightarrow f(w) \in B$$

dann ist auch A (semi-)entscheidbar.

Man schreibt diesfalls $A \leq_{\mathsf{m}} B$.

Satz von Rice

Sei R die Menge aller Turing-berechenbaren Funktionen und $S \subset R$ eine nicht-triviale Teilmenge ($\emptyset \neq S \neq R$) hiervon. Dann ist die Menge

$$L(S) = \{\langle M \rangle : M \text{ berechnet ein } f \in S\}$$

nicht entscheidbar.

Satz von Rice: Beispiel 1

Die Sprache VIRUS enthalte alle Programme, die bei leerer Eingabe die Ausgabe "`I LOVE YOU`" produzieren.

$$\text{VIRUS} = \{\langle M \rangle : M(\epsilon) = \text{“}\texttt{I LOVE YOU}\text{”}\}$$

Zeigen Sie: VIRUS ist nicht entscheidbar.

```
/**
 * Computes the mean of two integers.
 *
 * The result is rounded towards zero.
 *
 * @param n1  the first number
 *
 * @param n2  the second number
 *
 * @returns  floor((n1 + n2) / 2)
 *
 */
int mean(int n1, int n2) {
    // FIXME: This can overflow.
    return (n1 + n2) / 2;
}
```

Satz von Rice: Beispiel 2

Die Sprache EQUIV enthalte alle Programmdupel, die bei identischer Eingabe dieselbe Ausgabe produzieren.

$$\text{EQUIV} = \{(\langle M_1 \rangle, \langle M_2 \rangle) : \forall w \in \Sigma^* : M_1(w) = M_2(w)\}$$

Zeigen Sie: EQUIV ist nicht entscheidbar.

Kodierungsschema

Sei Π ein Problem und Σ ein Alphabet. Ein **Kodierungsschema** ist eine Abbildung

$$\begin{array}{rccc} s: & \Pi & \to & \Sigma^* \\ & I & \mapsto & \langle I \rangle \end{array},$$

die jeder Instanz I des Problems eine Kodierung $\langle I \rangle$ zuordnet.

Die *Größe* einer Probleminstanz I im Kodierungsschema s ist die Länge der Codierung $|s(I)|$.

Probleme

- Ein **Suchproblem** ist eine Frage nach einer Lösung.

  → Finde einen Spannbaum mit einem Gewicht von höchstens 42 für diesen Graphen.

- Ein **Optimierungsproblem** ist eine Frage nach einer (in einem gewissen Sinne) optimalen Lösung.

  → Finde einen Spannbaum minimalen Gewichts für diesen Graphen.

- Ein **Optimalwertproblem** ist eine Frage nach dem Wert einer (in einem gewissen Sinne) optimalen Lösung.

  → Welches Gewicht hat ein minimaler Spannbaum für diesen Graphen?

- Ein **Entscheidungsproblem** ist eine binäre Frage.

  → Existiert für diesen Graphen einen Spannbaum mit einem Gewicht von höchstens 42?

Entscheidungssprache

Sei Π ein Entscheiodungsproblem, Σ ein Alphabet und $s : \Pi \to \Sigma^*$ ein Kodierungsschema.

Die Sprache $L[\Pi, s]$ enthält alle in s kodierten Ja-Instanzen von Π.

Verifizierer

Seien Σ und Γ Alphabete, $L \subseteq \Sigma^*$ und V eine Zweiband-Turingmaschine mit Eingabealphabeten Σ und Γ.

V ist ein **Verifizierer** für L genau dann wenn für alle $w \in \Sigma^*$ gilt

$$w \in L \quad \Leftrightarrow \quad \exists c \in \Gamma^* : V \text{ akzeptiert } (w, c)$$

Ein solches c heißt *Zeuge* (oder *Zertifikat*) für w.

Zeugen

- Ist 618391 prim?
- Ist $947 \times 653 = 618391$ prim?

„947×653“ ist ein Zeuge für $618391 \in$ COMPOSITE.

Die Sprachklassen $\mathcal{P}$ und $\mathcal{NP}$

Die Klasse $\mathcal{P}$ enthält alle Sprachen, für die es einen **deterministischen Entscheider** (Turingmaschine) gibt, der für alle Worte der Länge n nach höchstens $p(n)$ Berechnungsschritten hält, für ein festes Polynom $p : \mathbb{N}_0 \to \mathbb{N}_0$.

Die Klasse $\mathcal{NP}$ enthält alle Sprachen, für die es einen **deterministischen Verifizierer** (bzw einen **nichtdeterministischen Entscheider**) gibt, der für alle Worte der Länge n nach höchstens $p(n)$ Berechnungsschritten hält, für ein festes Polynom $p : \mathbb{N}_0 \to \mathbb{N}_0$.