Theoretische Grundlagen der Informatik

Tutorium 11 — Wintersemester 2013/14

Moritz Klammler — 17. Dezember 2013

Organisatorisches

Das nächste Tutorium findet wieder am 7. Januar 2014 statt.

Tagesthemen

- Turing-Reduzierbarkeit
- Landau-Notation
- Lineares Speedup-Theorem
- Die Sprachklassen $\mathcal{P}$ und $\mathcal{NP}$

Nachtrag zum letzten Mal

In den Vorlesungsunterlagen steht:

> Die Menge der beweisbaren Aussagen in $\mathrm{Th}(\mathbb{N}, +, \times)$ ist rekursiv aufzählbar.
>
> Beweis: Zähle alle Strings auf, checke, ob es ein Beweis ist. Wenn ja, gib das Statement [den Satz, der bewiesen wird] aus.

Folgt daraus, dass $\mathrm{Th}(\mathbb{N}, +, \times)$ semientscheidbar ist?

Turing-Reduzierbarkeit

Sei Σ ein Alphabet, $L \subset \Sigma^*$ und O ein magisches Gerät (*Orakel*), das für jedes Wort $w \in \Sigma^*$ in konstanter Zeit entscheiden kann, ob $w \in L$. Eine Turingmaschine, deren Zustandsübergangsfunktion mitilfe von O berechnet wird, heißt *Orakel-Turingmaschine*.

Seien A und B Sprachen. A ist *Turing-reduzierbar* auf B genau dann wenn A von einer Orakel-Turingmaschine entschieden werden kann, die ein Orakel für B benutzt.

$$A \leq_{\mathsf{T}} B$$

Landau-Notation

Seien f und g Funktionen $\mathbb{N}_0 \to \mathbb{R}^+$.

Die Menge $\mathcal{O}(f)$ ist so definiert, dass

$$g \in \mathcal{O}(f) \quad \Leftrightarrow \quad \exists n \in \mathbb{N}_0 : \exists c \in \mathbb{R}^+ : \forall m \geq n : g(m) < cf(m)$$

Lineares Speedup-Theorem

Sei Σ ein Alphabet und $L \subset \Sigma^*$. Sei ferner $t : \mathbb{N}_0 \to \mathbb{N}$ eine monoton wachsende Funktion und M eine Turingmaschine, die L entscheidet und für jedes Wort $w \in \Sigma^*$ nach höchstens $t(|w|)$ Berechnungsschritten hält.

Dann gibt es für jedes $\alpha > 0$ eine Turingmaschine M', die L ebenfalls entscheidet, wobei es ein $n \in \mathbb{N}$ gibt, sodass M' für alle $w \in \Sigma^*$ mit $|w| \geq n$ nach höchstens $t'(|w|) = \alpha t(|w|) + |w| + 2$ Berechnungsschritten hält.

Beachte: $t \in \mathcal{O}(t')$

Welche Laufzeit hat dieser Algorithmus?

```
ROUTINE IsPrime(n : Integer)
BEGIN
    FOR m ← 2 TO n – 1 DO
        IF (n  mod m) = 0 THEN
            RETURN FALSE
        FI
    DONE
    RETURN TRUE
END
```

Welche Laufzeit hat dieser Algorithmus?

```
ROUTINE Max(a : Integer, b : Integer)
BEGIN
    IF a > b THEN
        RETURN a
    ELSE
        RETURN b
    FI
END
```

Verifizierer

Seien Σ und Γ Alphabete, $L \subset \Sigma^*$ 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 Akzeptor** (Turingmaschine) gibt, der für ein Wort der Länge n nach höchstens $\mathcal{O}(n^k)$ Berechnungsschritten hält für ein festes $k \in \mathbb{N}$.

Die Klasse $\mathcal{NP}$ enthält alle Sprachen, für die es einen **deterministischen Verifizierer** (bzw einen **nichtdeterministischen Akzeptor**) gibt, der für ein Wort der Länge n nach höchstens $\mathcal{O}(n^k)$ Berechnungsschritten hält für ein festes $k \in \mathbb{N}$.