**Theorie der Informatik**

G. Röger
Frühjahrssemester 2019

Universität Basel
Fachbereich Informatik

# Übungsblatt 9 — Lösungen

**Aufgabe 9.1** (Satz von Rice, 0.5+0.5+0.5+0.5 Punkte)

Bei welchen der folgenden Sprachen zeigt der Satz von Rice, dass die Sprache unentscheidbar ist? Geben Sie für Sprachen, bei denen der Satz von Rice verwendet werden kann, jeweils die Teilmenge von Turing-berechenbaren Funktionen $\mathcal{S}$ an, für die Sie den Satz anwenden.
*Hinweis:* Sie müssen keine Beweise angeben. Wenn der Satz von Rice anwendbar ist, geben Sie die Menge $\mathcal{S}$ an. Andernfalls geben Sie eine kurze Begründung (1 Satz) an, warum der Satz von Rice nicht anwendbar ist.

(a) $L = \{w \in \{0,1\}^* \mid M_w$ berechnet die zweistellige Multiplikationsfunktion$\}$

(b) $L = \{w \in \{0,1\}^* \mid$ Die Ausgabe von $M_w$ gestartet auf dem leeren Band enthält `0101`$\}$

(c) $L = \{w \in \{0,1\}^* \mid M_w$ hält für mindestens eine Eingabe nach mehr als 10 Schritten mit einer gültigen Ausgabe$\}$

(d) $L = \{w \in \{0,1\}^* \mid M_w$ berechnet eine zweistellige Funktion über den natürlichen Zahlen$\}$

**Lösung:**

(a) Der Satz von Rice ist mit der Funktionsmenge $\mathcal{S} = \{mul\}$ anwendbar.

(b) Der Satz von Rice ist mit der folgenden Funktionsmenge anwendbar:

$$\mathcal{S} = \{f \in \mathcal{R} \mid f(\varepsilon) \text{ enthält } \texttt{0101}\}$$

(c) Der Satz von Rice ist nicht direkt anwendbar, da es sich bei der Anzahl von Berechnungsschritten nicht um eine Eigenschaft der berechneten Funktion handelt.

*Anmerkung:* Ohne die Einschränkung der Schritte, also für

$$L' = \{w \in \{0,1\}^* \mid M_w \text{ hält für mindestens eine Eingabe mit einer gültigen Ausgabe}\},$$

wäre der Satz von Rice mit der Funktionsmenge $\mathcal{S} = \mathcal{R} \setminus \{\Omega\}$ anwendbar. Dabei ist $\Omega$ die überall undefinierte Funktion.

(d) Der Satz von Rice ist mit der Funktionsmenge $\mathcal{S} = \{f \in \mathcal{R} \mid f : \mathbb{N}_0^2 \to \mathbb{N}_0\}$ anwendbar.

**Aufgabe 9.2** (Nichtdeterministische Algorithmen, 2.5 Punkte)

Betrachten Sie das Entscheidungsproblem VertexCover:

- *Gegeben:* ungerichteter Graph $G = \langle V, E \rangle$, Zahl $K \in \mathbb{N}_0$
- *Gefragt:* Hat $G$ eine Knotenüberdeckung der Grösse höchstens $K$, also eine Menge von Knoten $S \subseteq V$ mit $|S| \leq K$ und $\{u, v\} \cap S \neq \emptyset$ für alle $\{u, v\} \in E$?

Geben Sie einen nichtdeterministischen Algorithmus für VertexCover an, dessen Laufzeit polynomiell in der Eingabegrösse ist. Begründen Sie, warum der Algorithmus korrekt ist und seine Laufzeit polynomiell ist.

**Lösung:**

Eingabe: Ungerichteter Graph $G = \langle V, E \rangle$ und $K \in \mathbb{N}_0$
$V' := V$
$S := \emptyset$
**WHILE** $|S| \neq K$ and $|S| \neq |V|$:
    **GUESS** $s \in V'$
    $V' = V' \setminus \{s\}$
    $S = S \cup \{s\}$
**FOR** $e \in E$:
    **IF** $e \cap S = \emptyset$:
        **REJECT**
**ACCEPT**

Das Verfahren ist korrekt: Zunächst wird eine Teilmenge $S \subseteq V$ mit $\max\{K, |V|\}$ Elementen geraten, die daraufhin überprüft wird, ob sie eine Knotenüberdeckung darstellt. Da jede beliebige solche Teilmenge geraten werden kann, kann man also eine tatsächliche Überdeckung raten, falls sie mit höchstens $K$ Knoten möglich ist. Diese besteht den Test und die entsprechende Berechnung akzeptiert. Gibt es hingegen keine solche Knotenüberdeckung, schlägt der Test unabhängig von der geratenen Teilmenge immer fehl.
Ausserdem benötigt das Verfahren nur polynomielle Laufzeit: Zum Raten der Teilmenge müssen wir die **WHILE**-Schleife $K$ mal durchlaufen (also linear oft). Da der GUESS-Schritt und das Einfügen und Löschen bei Mengen in polynomieller Zeit gehen, können wir die Teilmenge also in polynomieller Zeit raten. Für den Test, ob es sich tatsächlich um eine Knotenüberdeckung handelt, müssen wir linear oft (in $E$, was Teil der Eingabe ist) testen, ob zwei Mengen einen leeren Schnitt haben. Dies geht offensichtlich in polynomieller Zeit.

**Aufgabe 9.3** (Nichtdeterministische Algorithmen, 2.5+3 Punkte)
Betrachten Sie das Entscheidungsproblem CLIQUE:

- *Gegeben:* ungerichteter Graph $G = \langle V, E \rangle$, Zahl $K \in \mathbb{N}_0$
- *Gefragt:* Enthält $G$ eine Clique der Grösse $K$, also eine Menge von Knoten $C \subseteq V$ mit $|C| \geq K$ und $\{u, v\} \in E$ für alle $u, v \in C$ mit $u \neq v$?

(a) Geben Sie einen nichtdeterministischen Algorithmus für CLIQUE an, dessen Laufzeit durch ein Polynom in $|V| + |E|$ beschränkt ist. Begründen Sie, warum die Laufzeit des Algorithmus polynomiell ist.

**Lösung:**

Eingabe: Ungerichteter Graph $G = \langle V, E \rangle$ und $K \in \mathbb{N}_0$
$C = \emptyset$
**FOR** $v \in V$ :
    **GUESS** *take* $\in \{0, 1\}$
    **IF** *take* $== 1$ :
        $C = C \cup \{v\}$
**IF** $|C| < K$ :
    **REJECT**
**FOR** $v \in C$ :
    **FOR** $w \in C$ :
        **IF** $v \neq w$ and $\{v, w\} \notin E$ :
            **REJECT**
**ACCEPT**

Der obere Teil kann jede beliebige Teilmenge von Knoten $C \subseteq V$ raten. Der untere Teil verifiziert dann, dass es sich bei der geratenen Menge um eine Clique handelt. Wenn $G$ eine Clique der Grösse $K$ enthält, kann diese im oberen Teil geraten werden. Die geratene Menge übersteht dann alle Tests und wird akzeptiert. Wenn $G$ keine Clique der Grösse $K$ hat, führt jede Wahl von $C$ zu einem REJECT, entweder, weil $C$ weniger als $K$ Elemente hat, oder weil es zwischen zwei Elementen in $C$ keine Kante in $G$ gibt.

Jeder Schleifendurchlauf der ersten FOR-Schleife kann in konstanter Zeit implementiert werden, so dass die erste FOR-Schleife insgesamt Zeit $O(|V|)$ benötigt.

Der Test $|C| < K$ ist in konstanter Zeit möglich. Die nächsten beiden FOR-Schleifen können jeweils nicht öfter als $|C|$ mal durchlaufen werden und der Test in der inneren Schleife ist in konstanter Zeit möglich. Da $|C| \leq |V|$ ergibt sich damit für den zweiten Teil eine Laufzeit von $O(|V|^2)$.

Insgesamt beträgt die Laufzeit also $O(|V| + |V|^2) = O(|V|^2)$ ist also polynomiell.

(b) Geben Sie einen deterministischen Algorithmus für CLIQUE an und schätzen Sie seine Laufzeit in $O$-Notation ab.

**Lösung:**

Anstelle von GUESS-Anweisungen können wir systematisch alle Teilmengen ausprobieren, etwa in folgendem rekursiven Algorithmus:

```
FUNCTION compute-recursively(E, K, C, R) :
    IF R = ∅ : // test whether C is a clique
        IF |C| < K :
            RETURN False
        FOR v ∈ C :
            FOR w ∈ C :
                IF v ≠ w and {v, w} ∉ E :
                    RETURN False
        RETURN True
    ELSE :
        v = smallest element from R wrt. an arbitrary order on the vertices
        If compute-recursively(E, K, C ∪ {v}, R \ {v})
            RETURN True
        If compute-recursively(E, K, C, R \ {v})
            RETURN True
        RETURN False
```

Graph $G = (V, E)$ hat eine Clique der Grösse mindestens $L$, gdw. *compute-recursively*$(E, K, \{\}, V)$ true zurückgibt. Diese Berechnung geht systematisch alle möglichen Teilmengen von $V$ durch und testet, ob sie Cliquen in $G$ mit mindestend $K$ Elementen sind.

Sei $n = |V|$ die Anzahl der Knoten in $G$. Wie in Teil (a) ist die Laufzeit des Tests, ob eine gegebene Knotenmenge eine Clique ist, in $O(n^2)$.

Die Berechnung im ELSE-Fall benötigt jeweils höchstens Zeit $O(n)$ (für das Kopieren und Modifizieren der Mengen $C$ und $R$), wenn wir die Zeit für die rekursiven Funktionsaufrufe nicht mitrechnen.

Bei einer Eingabe mit $n$ Knoten werden insgesamt $2^n$ Teilmengen durchprobiert, und die rekursiven Funktionsaufrufe bilden einen vollständigen Binärbaum der Höhe $n$. Ein solcher Baum hat $2^n$ Blätter und $2^n - 1$ innere Knoten. Für jeden inneren Knoten wird der ELSE-Fall durchlaufen und für jedes Blatt der IF-Fall. Insgesamt ergibt sich damit die Laufzeit $O((2^n - 1)n + 2^n n^2) = O(n^2 2^n)$.