Muster-Beweise für Berechenbarkeit und Komplexität

von Niklas Rieken

In diesem Dokument wenden wir verschiedene Techniken an um Unberechenbarkeit von Funktionen bzw. Unentscheidbarkeit und NP-Vollständigkeit von Sprachen zu zeigen. Das Alphabet sei im gesamten Dokument einfach $\Sigma = \{0, 1\}$.

Berechenbarkeit

Wir zeigen für eine Funktion, dass sie nicht berechenbar ist. In der Vorlesung und Übung wurden lediglich Sprachen, d.h. Funktionen der Form $\{0,1\}^* \to \{0,1\}$ betrachtet. Die Herangehensweise bei allgemeineren Funktionen ist jedoch ziemlich gleich und es ist für das Verständnis des Stoffes vermutlich auch ganz gut einen solchen Beweis einmal gesehen zu haben.
Wir benutzen einmal den Satz von Rice und alternativ dazu die (Turing-)Reduktion und zuletzt einmal Unterprogrammtechnik. Gegeben sei die Funktion

$$s : \Sigma^* \to \Sigma^*, s(x) = \begin{cases} \langle \mathcal{A} \rangle & x = \langle M \rangle \text{ und } L(M) = L(\mathcal{A}) \text{ regulär} \\ x & x = \langle M \rangle \text{ und } L(M) \text{ nicht regulär} \\ 0 & \text{sonst,} \end{cases}$$

wobei $\mathcal{A}$ ein DFA ist, der die selbe Sprache erkennt wie M und $\langle \mathcal{A} \rangle$ eine geeignete (d.h. eindeutige und von TM-Codierung unterscheidbare) DFA-Codierung.
Zeige: s ist unberechenbar.
Beweis. Wir zeigen zunächst folgende Hilfsaussage. Die Sprache

$$K = \{\langle M \rangle \mid L(M) \text{ ist endlich}\}$$

ist unentscheidbar.
Wir betrachten dazu einmal diese Sprache K mit dem Satz von Rice: Sei

$$\mathcal{S} = \{f_M \mid \text{ es ex. ein } w_0 \in \Sigma^* \text{ s.d. f.a. } w \text{ mit } |w| \geq |w_0| \text{ gilt: } f_M(w) \notin 1\Sigma^*\}.$$

$\mathcal{S}$ ist nicht trivial. Betrachte dazu die berechenbaren Funktionen:

$$f_i : \Sigma^* \to \Sigma \text{ mit } f_i(w) = i \quad \text{f.a. } w \in \Sigma^*$$

mit $i \in \{0, 1\}$. Dann ist

- $f_0 \in \mathcal{S}$ und
- $f_1 \notin \mathcal{S}$

und damit $\varnothing \subset \mathcal{S} \subset \mathcal{R}$. Nach Satz von Rice ist dann

$$\begin{aligned} L(\mathcal{S}) &= \{\langle M \rangle \mid M \text{ berechnet eine Funktion aus } \mathcal{S}\} \\ &= \{\langle M \rangle \mid M \text{ antw. auf Eingaben ab einer bestimmten Länge immer mit } 0\} \\ &= \{\langle M \rangle \mid M \text{ antw. nur auf endlich vielen Eingaben mit } 1\} \\ &= \{\langle M \rangle \mid L(M) \text{ ist endlich}\} \\ &= K \end{aligned}$$

unentscheidbar. ■

Um auch die Reduktion einmal zu üben werden wir die selbe Hilfsaussage auch damit einmal beweisen.
Wir konstruieren eine berechenbare Funktion $f : \Sigma^* \to \Sigma^*$, sodass gilt $x \in \overline{H_\varepsilon}$ g.d.w. $f(x) \in K$, wobei $\overline{H_\varepsilon}$ das Komplement des gewohnten speziellen Halteproblems ist. Wir konstruieren f wie folgt:

- Falls x keine korrekte Codierung einer TM ist, setze $f(x) = \langle M_0 \rangle$, wobei M_0 die Funktion f_0 berechnet.
- Andernfalls sei $x = \langle M \rangle$ und wir konstruieren eine TM M^* wie folgt:
  1. M^* bestimmt die Eingabelänge n
  2. M^* löscht die Eingabe.
  3. M^* simuliert M auf ε, aber höchstens n Schritte.
  4. Falls die Simulation von M gehalten hat, akzeptiert M^*, sonst verwirft sie.

  Wir setzen dann $f(x) = \langle M^* \rangle$.

Die Funktion f ist berechenbar, denn (1. Bullet) wir können natürlich für Wörter entscheiden, ob es TM-Codierungen sind (regulär) und ggf. auf eine einfache TM abbilden. Außerdem können wir (2. Bullet) als "Preprocessing" Eingabelängen bestimmen und Eingaben löschen (1, 2, siehe Übung), TMs simulieren (3, universelle TM) und danach abhängig vom zuletzt besuchten Zustand der Simulation eine Ausgabe machen (4, klar).
Zur Korrektheit: Falls x keine TM-Codierung war, so ist $x \in \overline{H_\varepsilon}$ und $f(x) = \langle M_0 \rangle \in K$.
Sonst betrachte wieder $x = \langle M \rangle$.

$$
\begin{aligned}
x \in \overline{H_\varepsilon} &\implies M \text{ hält nicht auf } \varepsilon \\
&\implies \text{es gibt kein } i, \text{ sodass } M \text{ nach spätestens } i \text{ Schritten hält} \\
&\implies \text{es gibt keine Eingabe } w, \text{ sodass } M^* \text{ akzeptiert} \\
&\implies \text{auf allen Eingaben verwirft } M^* \text{ (sie hält ja immer)} \\
&\implies L(M^*) = \varnothing \\
&\implies L(M^*) \text{ ist endlich} \\
&\implies f(x) = \langle M^* \rangle \in K.
\end{aligned}
$$

$$
\begin{aligned}
x \notin \overline{H_\varepsilon} &\implies M \text{ hält auf } \varepsilon \\
&\implies \text{es gibt ein } i, \text{ sodass } M \text{ nach spätestens } i \text{ Schritten hält} \\
&\implies \text{es gibt ein } w_0, \text{ sodass } M \text{ auf allen } w \text{ mit } |w| \geq |w_0| \text{ sind, hält} \\
&\implies \text{es gibt ein } w_0, \text{ sodass } M^* \text{ auf allen } w \text{ mit } |w| \geq |w_0| \text{ akzeptiert} \\
&\implies \text{auf fast allen Eingaben akzeptiert } M^* \\
&\text{("fast alle" meint "auf nur endlich vielen nicht")} \\
&\implies L(M^*) \text{ ist unendlich} \\
&\implies f(x) = \langle M^* \rangle \notin K.
\end{aligned}
$$

Damit gilt $\overline{H_\varepsilon} \leq K$. Da $\overline{H_\varepsilon}$ unentscheidbar folgt auch, dass K unentscheidbar ist. ■

Wir kommen nun zum Beweis, dass s nicht rekursiv ist. Dazu verwenden wir die Unterprogrammtechnik.

Wir nehmen zum Widerspruch an, dass s berechenbar wäre. Also existiert eine Turingmaschine M_s die s berechnet. Wir konstruieren eine Turingmaschine M_K, die M_s als Unterprogramm verwendet und dann K entscheiden würde. Eine Skizze der Konstruktion ist in Abbildung 1.
Die TM M_K arbeitet wie folgt:

- Eingaben x die keine TM-Codierung sind werden direkt verworfen.
- Falls $x = \langle M \rangle$ rufe M_s als Unterprogramm auf $\langle M \rangle$ auf.
  - Falls M_s die Ausgabe 0 ausgibt oder eine TM-Codierung, verwirft M_K
  - Sonst rufe ein weiteres Unterprogramm D mit $\langle \mathcal{A} \rangle$ auf, welches zu einer als DFA gegebenen regulären Sprache entscheidet ob sie endlich ist und übernehme dessen Ausgabe.
    (Bemerkung: Das Problem ist entscheidbar. Erhält man die Sprache als DFA, so kann man überprüfen ob es im Transitionsgraphen vom Startzustand einen Pfad zu einem akzeptierenden Zustand gibt mit Zustandswiederholung (vgl. Pumping-Lemma))

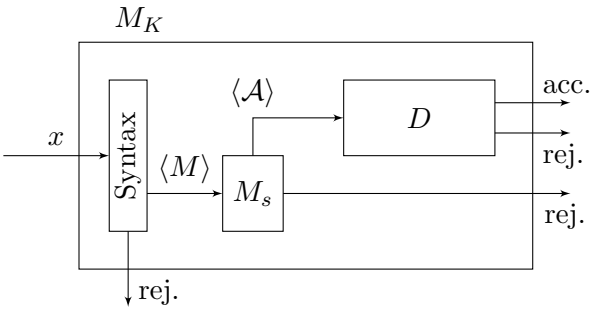

Abbildung 1: Beweisskizze

Zur Korrektheit: Falls x keine TM-Codierung ist, so ist $x \notin K$ und M_K verwirft. Sei nun $x = \langle M \rangle$.

$$
\begin{aligned}
x \in K &\implies L(M) \text{ ist endlich} \\
&\implies L(M) \text{ ist regulär (da alle endlichen Sprachen regulär sind)} \\
&\implies M_s \text{ gibt DFA } \langle \mathcal{A} \rangle \text{ aus} \\
&\implies D \text{ akzeptiert } \langle \mathcal{A} \rangle \\
&\implies M_K \text{ akzeptiert } \langle M \rangle = x.
\end{aligned}
$$

$$
\begin{aligned}
x \notin K &\implies L(M) \text{ ist unendlich} \\
&\implies (L(M) \text{ ist regulär}) \text{ oder } (L(M) \text{ ist nicht regulär}) \\
&\implies (M_s \text{ gibt DFA } \langle \mathcal{A} \rangle \text{ aus}) \text{ oder } (M_s \text{ gibt TM-Codierung aus}) \\
&\implies (D \text{ verwirft}) \text{ oder } (M_K \text{ verwirft}) \\
&\implies M_K \text{ verwirft } \langle M \rangle = x.
\end{aligned}
$$

Insgesamt folgt also, dass M_K akzeptiert x g.d.w. $x = \langle M \rangle$ und $L(M)$ endlich. Damit entscheidet M_K die Sprache K, welche aber unentscheidbar ist nach unser Hilfsaussage. Das ist ein Widerspruch und es folgt, dass M_s nicht existieren kann. Also ist auch s nicht berechenbar. □

Komplexität

Das Prinzip der polynomiellen Reduktion (auch Cook-Reduktion) ist quasi das selbe wie die der Turing-Reduktion im vorigen Abschnitt. Der Unterschied besteht allein darin, dass die polynomielle Reduktion eine polynomielle Laufzeit haben muss.
Wir betrachten das folgende Entscheidungsproblem:

IN-STABLE
Input: gerichteter Graph $G = (V, E)$, ein Knoten $v \in V$.
Output: Ja, g.d.w. ein von Neumann-Morgenstern Stable Set U existiert mit $v \in U$.

Zunächst klären wir was ein von Neumann-Morgenstern Stable Set (kurz: Stable Set) ist: In einem Graphen $G = (V, E)$ ist eine Knotenteilmenge $U \subseteq V$ ein Stable Set, wenn zwei Bedingungen erfüllt sind:

innere Stabilität Für alle $u \neq v \in U$ gilt $(u, v) \notin E$.

äußere Stabilität Für alle $u \notin U$ existiert ein $v \in U$ mit $(v, u) \in E$.

Man kann auch formilieren, dass ein Stable Set eine Graphen G sowohl ein Independent Set (innere Stabilität) als auch ein Dominating Set (äußere Stabilität) ist. Beachte, dass unser Ziel nicht ist, wie bei vielen Problemen der Übung, ein möglichst kleines solches U zu finden, sondern für einen gegebenen Knoten ein solches Set zu finden. Im Gegensatz zu den genannten Problemen oben muss dieses nicht existieren (Independent Set: beliebiges Singleton, Dominating Set: alle Knoten). In Abbildung 2 ist ein Stable Set gezeichnet.

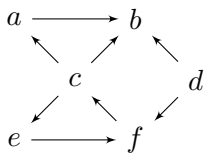

Abbildung 2: Stable Set $U = \{c, d\}$.

Wir zeigen, dass das Problem IN-STABLE NP-vollständig ist.
Zugehörigkeit zu NP ist einfach einzusehen. Die Eingabe sei $(G = (V, E), v)$. Als Zeugen wählen wir die das Stable Set U. Als Teilmenge der Knoten hat U (z.B. binär codiert) auch nur polynomielle Länge. Der Verifizierer arbeitet wie folgt:

1. Prüfe, ob $v \in U$.
2. Prüfe, ob U ein Stable Set ist, d.h. im Detail:
   - Prüfe ob für alle Knoten $u, u' \in U$, die Kanten (u, u') und (u', u) nicht existieren.
   - Prüfe ob für alle Knoten $u \notin U$ ein Knoten $u' \in U$ existiert mit Kante $(u', u) \in E$.

Schritt 1 kann offensichtlich in linearer Zeit durchgeführt werden. Für beide Teilschritte aus Schritt 2 prüfen wir für je $\mathcal{O}(n^2)$ viele Knotenpaare die maximal $\mathcal{O}(n^2)$ vielen Kanten. Also hat der Verifizierer eine polynomielle Laufzeit. ■

Für den Beweis der NP-Härte zeigen wir eine polynomielle Reduktion von SATISFIABILITY auf IN-STABLE. Sei dazu $\varphi := \bigwedge_{i=1}^{m} \bigvee_{j=1}^{k_i} \ell_{ij}$ eine aussagenlogische Formel in konjunktiver

Normalform mit Literalen $\ell_{ij} \in \{x_1, \bar{x}_1, \ldots, x_n, \bar{x}_n\}$, wobei $\bar{x}_i$ dem Literal $\neg x_i$ entspricht. φ hat m Klauseln und k_i gibt die Anzahl der Literale in der i-ten Klausel (im Folgenden c_i) an. Aus φ konstruieren wir nun eine Instanz für In-Stable, also einen Graphen $G = (V, E)$ und eine ausgewählten Knoten $v \in V$. Sei dafür

$$\begin{aligned} V := &\{c_{i1}, c_{i2}, c_{i3} \mid 1 \leq i \leq m\} \\ &\cup \{x_i, \bar{x}_i, x'_i, \bar{x}'_i : 1 \leq i \leq n\} \\ &\cup \{d_1, d_2, d_3, d_4\}, \end{aligned}$$

d.h. für jede Klausel aus φ gibt es drei Knoten c_{i1}, c_{i2}, c_{i3}, für jede Variable aus φ vier Knoten $x_i, \bar{x}_i, x'_i, \bar{x}'_i$ und zusätzlich noch vier weitere Knoten d_1, d_2, d_3, d_4, diese benötigen wir um die Existenz eines Stable Sets zu garantieren. Für d_1 soll am Ende gelten, dass d_1 zu einem Stable Set U gehört, genau dann, wenn φ erfüllbar ist (d.h. $v = d_1$ für die In-Stable-Instanz). Die Kantenrelation E konstruieren wir wie folgt: Für jedes Klausel-Tripel bildet E einen gerichteten Kreis der Länge 3. Die Quadrupel für jede Variable x_i bilden einen gerichteten Kreis der Länge 4 bezüglich E, ebenso die d_i. Für jedes Literal $\ell_{ij} \in \{x_k, \bar{x}_k : 1 \leq k \leq n\}$ – als Knoten aufgefasst – gelte außerdem, dass $(l_{ij}, c_{ik}) \in E, k \in \{1, 2, 3\}$. Zuletzt fügen wir die Kanten (d_2, c_{ik}) für jede Klausel c_i und ihre drei $(k = 1, 2, 3)$ Knoten .
Formal fassen wir dies zusammen:

$$\begin{aligned} E := &\left\{(c_{i1}, c_{i2}), (c_{i2}, c_{i3}), (c_{i3}, c_{i1}) \,\middle|\, 1 \leq i \leq m\right\} \\ &\cup \left\{(x_i, \bar{x}_i), (\bar{x}_i, x'_i), (x'_i, \bar{x}'_i), (\bar{x}'_i, x_i) \,\middle|\, 1 \leq i \leq n\right\} \\ &\cup \{(d_1, d_2), (d_2, d_3), (d_3, d_4), (d_4, d_1)\} \\ &\cup \bigcup_{j=1}^{m} \bigcup_{r=1}^{k_j} \{(\ell_{jr}, c_{j1}), (\ell_{jr}, c_{j2}), (\ell_{jr}, c_{j3})\} \\ &\cup \left\{(d_2, c_{ij}) \,\middle|\, 1 \leq i \leq m, j \in \{1, 2, 3\}\right\}. \end{aligned}$$

In Abbildung 3 ist für ein moderates φ der entsprechend resultierende Graph einmal dargestellt. Aus der Konstruktion von V und E können wir einige Schlussfolgerungen ziehen:

1. Die Menge $\{x_i, x'_i \mid 1 \leq i \leq n\} \cup \{d_2, d_4\}$ ist für jedes φ ein Stable Set.

2. Jedes Stable Set enthält entweder d_1, d_3 oder d_2, d_4, aufgrund der inneren Stabilität aber niemals beide Paare.

3. Jedes Stable Set enthält für jedes $i \in \{1, \ldots, n\}$ entweder x_i, x'_i oder $\bar{x}_i, \bar{x}'_i$, jedoch ebenfalls wegen der internen Stabilität niemals beide Paare.

4. Zwei Knoten c_{ij}, c_{ik} mit $j \neq k$, die aus den Klauseln von φ generiert wurden, können nicht zusammen in einem Stable Set liegen. Daraus folgt aber, dass es für jede Klausel einen Knoten geben muss, die zwei Knoten der zugehörigen Klausel dominieren. Da jedoch jeder Knoten, der eine Kanten zu einem Knoten c_{ij} der Klausel c_i hat, zu allen Knoten von c_i eine Kanten haben muss, folgt, dass kein Knoten, der aus einer Klausel generiert wurde, in einem Stable Set liegen kann.

Nun schließen wir die Reduktion: Ein Stable Set muss entweder d_2 enthalten oder eine Teilmenge der Knoten, die aus den Literalen generiert werden, wobei für jede Klausel c_i ihre zugehörigen Knoten $c_{ij}, j \in \{1, 2, 3\}$ von einem Knoten x_k oder $\bar{x}_k$ eine eingehende Kante haben müssen. Die korrespondierenden Literale sind nach der Konstruktion natürlich in der Klausel.
Wegen (3) gilt allerdings, dass nicht beide Knoten $x_k, \bar{x}_k$ im Stable Set enthalten sind, also

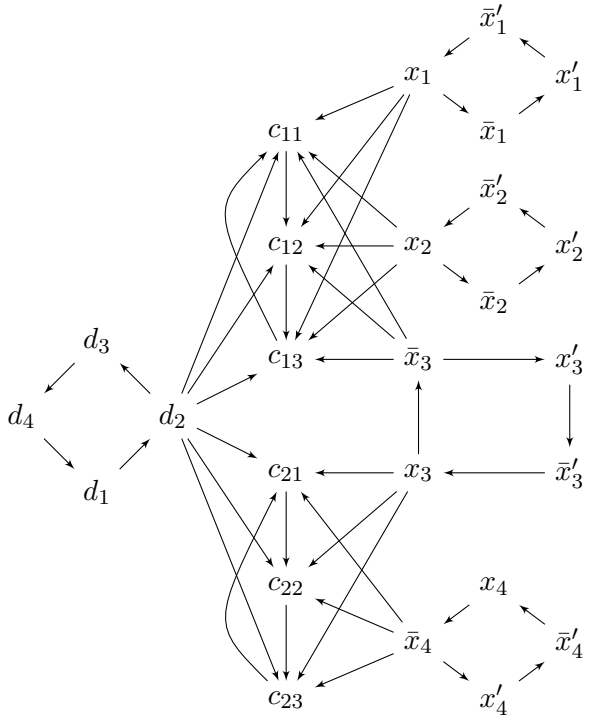

Abbildung 3: Graph aus der Konstruktion für die aussagenlogische Formel $\varphi = (x_1 \vee x_2 \vee \neg x_3) \wedge (x_3 \vee \neg x_4)$.

liefert das Stable Set eine erfüllende Belegung für φ. Mit (2) folgt, dass d_1 genau dann zu einem Stable Set gehört, wenn φ erfüllbar ist.
V und E können in polynomieller Zeit berechnet werden, da wir für jede Klausel und jede Variable nur konstanten Aufwand haben und damit ist die NP-Vollständigkeit gezeigt. □