Beweistechniken: Entscheidbarkeit und Unentscheidbarkeit

Holger Dell

5. Februar 2024

Übliche Klausuraufgabe

Die meisten Klausuraufgaben zu Entscheidbarkeit sehen so oder ähnlich aus:

> Sei L die folgende Sprache über dem Alphabet Σ:
>
> $$L = \Big\{ x \in \Sigma^* \;\Big|\; \text{irgendeine Bedingung an } x \Big\}.$$
>
> Ist L entscheidbar oder unentscheidbar? Beweise deine Behauptung.

Natürlich wird statt „irgendeine Bedingung an x“ eine konkrete Bedingung angegeben. Auch kann die Eingabe x anders formatiert sein, zum Beispiel kann die Eingabe auch $\langle M \rangle$ für eine Turingmaschine M, oder $\langle M, w \rangle$ für eine Turingmaschine M und ein Wort w sein.

1 Entscheidbarkeit

Um zu beweisen, dass die Sprache entscheidbar ist, muss ein deterministischer Algorithmus angegeben werden, der die Sprache entscheidet. Zum Beispiel:

$$L_1 = \Big\{ x \in \{0,1\}^* \;\Big|\; x \text{ ist die Binärkodierung einer geraden Zahl} \Big\}$$

Die Sprache L_1 ist entscheidbar, da folgender Algorithmus die Sprache entscheidet:

> **Eingabe:** Wort $x \in \{0,1\}^*$
> **Algorithmus:** Laufe den Lesekopf zum rechtesten Bit der Eingabe x. Falls es eine 0 ist, gib „Ja“ aus. Falls es eine 1 ist, gib „Nein“ aus.

2 Unentscheidbarkeit

Falls die Sprache unentscheidbar ist, so gibt es mehrere mögliche Ansätze, dies zu beweisen: Satz von Rice, Reduktion, Widerspruch.

2.1 Beweis durch Satz von Rice

Man kann den Satz von Rice nicht immer anwenden. Zum Beispiel kann man den Satz von Rice nicht oder nicht direkt anwenden, wenn die Instanzen nicht das Format $\langle M \rangle$ haben. Man kann den Satz von Rice auch nicht auf die Sprache SelfHalt anwenden, da die Eigenschaft „Maschine M hält auf Eingabe $\langle M \rangle$“ eine selbstreferentielle Eigenschaft der Maschine M ist und sich nicht nur auf die von M berechnete Funktion bezieht.

Falls der Satz von Rice angewendet werden kann, dann ist diese Beweismethode in den allermeisten Fällen der einfachste Ansatz, da man relativ mechanisch vorgehen kann, sofern man die Methode verinnerlicht hat. Zum Beispiel:

$$L_1 = \left\{ \langle M \rangle \mid M \text{ ist eine Turingmaschine, die genau eine Eingabe akzeptiert} \right\}$$

L_1 ist ein Problem, bei dem die Probleminstanzen von der Form $\langle M \rangle$ sind, und das sich nicht auf die syntaktischen Eigenschaften von M bezieht, sondern nur auf die Funktion, die von M berechnet wird. Daher kann der Satz von Rice angewendet werden.

1. **(Ganz wichtig!) Genaue Definition von** S: Wir müssen die Menge S definieren, die alle partiellen Funktionen enthält, die von Maschinen in L_1 berechnet wird. In unserem Fall definieren wir S als die Menge aller berechenbaren Funktionen

   $$f : \Sigma^* \to \{\texttt{accept}, \texttt{reject}, \texttt{undefined}\},$$

   die auf genau einer Eingabe `accept` ausgeben:

   $$S = \left\{ f : \Sigma^* \to \{\texttt{accept}, \texttt{reject}, \texttt{undefined}\} \;\middle|\; \begin{array}{c} f \text{ ist berechenbar und} \\ f(w) = \texttt{accept} \text{ für genau ein } w \in \Sigma^* \end{array} \right\}$$

2. **Korrektheit der Definition von** S: Der Satz von Rice macht nur Aussagen über die Sprache $C(S)$. Daher müssen wir zeigen, dass $L_2 = C(S)$ gilt. Hierzu zeigen wir beide Inklusionen $L_2 \subseteq C(S)$ und $L_2 \supseteq C(S)$:
   - Sei $\langle M \rangle \in L_2$. Dann ist M eine Turingmaschine, die genau eine Eingabe akzeptiert. Daher berechnet M eine partielle Funktion $f : \Sigma^* \to \{\texttt{accept}, \texttt{reject}, \texttt{undefined}\}$, die auf genau einer Eingabe `accept` ausgibt. Daher gilt $f \in S$ und somit $\langle M \rangle \in C(S)$.
   - Sei umgekehrt $f \in S$. Dann ist f eine partielle Funktion, die von einer Turingmaschine M berechnet wird, die genau eine Eingabe akzeptiert. Daher gilt $\langle M \rangle \in L_2$.

3. **(Einfach, aber wichtig!) Beweis, dass** S **nicht trivial ist**: Wir müssen beweisen, dass S nicht die Menge aller berechenbaren partiellen Funktionen ist und auch nicht die leere Menge. Wir definieren hierzu zwei partielle Funktionen f_1 und f_2 wie folgt:
   - f_1 ist die partielle Funktion, die genau eine Eingabe akzeptiert, nämlich die Eingabe 0. Also $f_1(0) = 1$ und $f_1(w)$ ist undefiniert falls $w \in \Sigma^*$ mit $w \neq 0$. Diese Funktion wird von einer Turingmaschine berechnet, die Eingabe 0 akzeptiert und alle anderen Eingaben verwirft. Daher gilt $f_1 \in S$.
   - f_2 ist die partielle Funktion Ω, die überall undefiniert ist. Diese Funktion wird von der Turingmaschine berechnet, die auf keiner Eingabe hält. Daher ist f_2 eine berechenbare partielle Funktion mit $f_2 \notin S$.

   Wir haben gezeigt, dass $L_2 = C(S)$ für eine nicht-triviale Menge S von berechenbaren partiellen Funktionen gilt. Laut Satz von Rice ist L_2 dann unentscheidbar.

2.2 Beweis durch Reduktion

Falls der Satz von Rice nicht anwendbar ist, könnte eine Reduktion funktionieren. Hierzu suchen wir uns eine Sprache, die wir bereits als unentscheidbar kennen, und konstruieren eine Reduktion von dieser Sprache auf die Sprache, von der wir zeigen wollen, dass sie unentscheidbar ist. Zum Beispiel:

$$\textsc{Halt} = \left\{ \langle M, w \rangle \mid M \text{ ist eine Turingmaschine, die auf Eingabe } w \text{ hält} \right\}$$

Hierzu eignet sich als Reduktionspartner die Sprache Accept, von der wir bereits gezeigt haben, dass sie unentscheidbar ist:

$$\textsc{Accept} = \left\{ \langle M, w \rangle \mid M \text{ ist eine Turingmaschine, die } w \text{ akzeptiert} \right\}.$$

(Ganz wichtig!) Präzise Beschreibung der Reduktion, z.B. als Algorithmus. Wir konstruieren eine Reduktion R von Accept auf Halt. R erhält als Eingabe eine Instanz $\langle M, w \rangle$ für das Problem Accept und geht wie folgt vor:

1. R berechnet eine Instanz $\langle M', w' \rangle$ für das Problem Halt wie folgt:
2. R setzt $w' = w$ und konstruiert eine Turingmaschine M', die genau wie M arbeitet, jedoch wie folgt verändert wird: immer wenn M verwerfen würde, läuft M' stattdessen in eine Endlosschleife.
3. R gibt $\langle M', w \rangle$ aus.

(Wichtig!) Korektheitsbeweis für die Reduktion. Wir müssen nun zeigen, dass R eine Reduktion von Accept auf Halt ist. Hierzu zeigen wir, dass R korrekt ist:

- Sei $\langle M, w \rangle \in \textsc{Accept}$. Dann akzeptiert M die Eingabe w. Daher akzeptiert M' die Eingabe w und hält. Also gilt $\langle M', w \rangle \in \textsc{Halt}$.
- Sei umgekehrt $\langle M, w \rangle \notin \textsc{Accept}$. Dann verwirft M die Eingabe w oder hält nicht auf Eingabe w. Falls M die Eingabe w verwirft, läuft M' auf Eingabe w in eine Endlosschleife. In beiden Fällen gilt also $\langle M', w \rangle \notin \textsc{Halt}$.

Wir haben gezeigt, dass R eine Reduktion von Accept auf Halt ist. Da Accept unentscheidbar ist, muss auch Halt unentscheidbar sein.

2.3 Beweis durch Widerspruch

Der vermutlich schwierigste Ansatz ist es, den Widerspruchsbeweis für die Unentscheidbarkeit des speziellen Halteproblems SelfHalt zu imitieren. Das müssen wir immer dann tun, wenn der Satz von Rice nicht anwendbar ist und wir auch keine Reduktion finden können. Dazu nehmen wir an, dass die Sprache entscheidbar ist, und führen dies zu einem Widerspruch. Zum Beispiel:

$$L_3 = \left\{ \langle M \rangle \mid \langle M \rangle \text{ ist die Kodierung einer Turingmaschine, die auf Eingabe } \langle M \rangle \langle M \rangle \text{ hält} \right\}$$

Um zu beweisen, dass L_2 unentscheidbar ist, nehmen wir an, dass L_2 entscheidbar ist, und führen diese Annahme zum Widerspruch.

1. **Annahme**: Wir nehmen an, dass L_2 entscheidbar ist. Das bedeutet, es gibt eine Turingmaschine A, die für jede Eingabe entscheiden kann, ob sie zu L_2 gehört oder nicht.

2. **Konstruktion von B**: Wir konstruieren eine neue Turingmaschine B. Diese Turingmaschine B nimmt als Eingabe ein Wort $x \in \Sigma^*$ und arbeitet wie folgt:
   - B prüft zunächst, ob x geschrieben werden kann als $x = yy$ für ein Wort $y \in \Sigma^*$. Falls das nicht der Fall ist, dann hält B in einem beliebigen Zustand.
   - Falls $x = yy$ gilt, dann wendet B die Maschine A auf y an.
   - Wenn A entscheidet, dass $y \in L_2$, dann geht B in eine Endlosschleife und hält nicht.
   - Wenn A entscheidet, dass $y \notin L_2$, dann hält B.

3. **Widerspruch**: Nun beweisen wir, dass weder $\langle B \rangle \in L_2$ noch $\langle B \rangle \notin L_2$ gilt, was absurd ist und also zu einem Widerspruch zur Annahme führt.
   - Wenn $\langle B \rangle \in L_2$, dann hält B gemäß Definition von L_2 auf der Eingabe $\langle B \rangle \langle B \rangle$. Aber laut der Definition von B würde B bei Eingabe $\langle B \rangle \langle B \rangle$ zunächst A auf Eingabe $\langle B \rangle$ aufrufen; A entscheidet, dass $\langle B \rangle \in L_2$ gilt, und dann geht B in eine Endlosschleife, was ein Widerspruch ist. Also kann $\langle B \rangle \in L_2$ nicht gelten.
   - Wenn umgekehrt $\langle B \rangle \notin L_2$, dann hält B gemäß Definition von L_2 auf der Eingabe $\langle B \rangle \langle B \rangle$ nicht. Aber laut der Definition von B würde B bei Eingabe $\langle B \rangle \langle B \rangle$ zunächst A auf Eingabe $\langle B \rangle$ aufrufen; A entscheidet, dass $\langle B \rangle \notin L_2$ gilt, und dann B würde halten, was ein Widerspruch ist. Also kann $\langle B \rangle \notin L_2$ nicht gelten.

4. **Schlussfolgerung**: In beiden Fällen führt die Annahme, dass L_2 entscheidbar ist, zu einem Widerspruch. Daher ist L_2 unentscheidbar.

Dieser Beweis nutzt die Selbstreferenz und die Idee des Widerspruchs, um zu zeigen, dass es keine Turingmaschine geben kann, die konsistent entscheiden kann, ob eine andere Turingmaschine auf einer gegebenen Eingabe hält oder nicht.

Beachte, dass man auch per Reduktion zeigen kann, dass L_3 unentscheidbar ist. Beispielsweise kann mit folgender Reduktionsfunktion von SelfHalt auf L_2 reduziert werden: Bei Eingabe $\langle M \rangle$, konstruiere eine neue Maschine M', die ihre Eingabe ignoriert und die Maschine M auf Eingabe $\langle M \rangle$ ausführt. Diese Reduktionsfunktion ist berechenbar und es gilt:

$$\begin{aligned}
\langle M \rangle \in \text{SelfHalt} &\iff M \text{ hält auf Eingabe } \langle M \rangle \\
&\iff M' \text{ hält auf allen Eingaben} \\
&\iff M' \text{ hält auf Eingabe } \langle M' \rangle \langle M' \rangle \\
&\iff M' \in L_3.
\end{aligned}$$