Theoretische Informatik für Wirtschaftsinformatik und Lehramt

Entscheidungsprobleme

Priv.-Doz. Dr. Stefan Milius
stefan.milius@fau.de

Theoretische Informatik
Friedrich-Alexander Universität Erlangen-Nürnberg

SS 2016

Gliederung

Worum geht es in diesem Abschnitt?

- Entscheidungsprobleme

  Probleme mit nur zwei möglichen Lösungen:
  „Ja“ oder **„Nein“**

  Frage: Gibt es für jedes solche Problem einen Algorithmus?

- Berühmtes **unentscheidbares** Problem: Halteproblem

  Kann ein Algorithmus feststellen, ob ein gegebenes Programm terminiert oder nicht?

- Damit lassen sich weitere Entscheidungsprobleme als unentscheidbar nachweisen.

Lernziele

- die Konzepte Entscheidbarkeit, Unentscheidbarkeit, Semi-Entscheidbarkeit beispielbezogen erklären können
- die Unentscheidbarkeit des Halteproblems erläutern können
- Methoden für Unentscheidbarkeitsbeweise:
  1. Satz von Rice anwenden können
  2. Sprache auf eine andere Sprache reduzieren können

Entscheidungsprobleme (I)

Entscheidungsprobleme sind Probleme, die nur zwei mögliche Lösungen haben: „Ja“ oder „Nein“.

Beispiel: Primzahlproblem

Eingabe: Natürliche Zahl $n \in \mathbb{N}$
Aufgabe: Entscheiden, ob n eine Primzahl ist.

Bemerkung: Der „Problembegriff“ wird verwendet für eine ganze Klasse gleichartiger Einzelprobleme:

„Ist 1 eine Primzahl?“, „Ist 2 eine Primzahl?“, ...

Beispiel: Wortproblem einer formalen Sprache

Zu jeder formalen Sprache $L \subseteq \Sigma^*$ ist das zugehörige **Wortproblem**:
Eingabe: $w \in \Sigma^*$
Aufgabe: Entscheiden, ob $w \in L$ oder nicht.

Modellierung von Entscheidungsproblemen

Durch geeignete Kodierung lässt sich jedes Entscheidungsproblem in ein Wortproblem umwandeln.

Beispiel

- Kodierung natürlicher Zahlen als Wörter über $\Sigma = \{0, \ldots, 9\}$ (Dezimaldarstellung)
- Primzahlen bilden dann eine formale Sprache:

$$\mathit{PRIMES} \subseteq \Sigma^*$$

- Primzahlproblem = Wortproblem für *PRIMES*.

Entscheidungsverfahren und Semi-Entscheidungsverfahren

Im Folgenden: Algorithmen für Entscheidungsprobleme

Man unterscheidet:

- **Entscheidungsverfahren**:

  Algorithmus, der für alle Eingaben terminiert und die korrekte Antwort „Ja“ oder „Nein“ liefert

- **Semi-Entscheidungsverfahren**:

  Algorithmus, der für jede Eingabe, für die die korrekte Antwort „Ja“ lautet, mit der korrekten Antwort „Ja“ terminiert und der für jede Eingabe, für die die korrekte Antwort „Nein“ lautet, entweder mit der Antwort „Nein“ oder gar nicht terminiert

Entscheidbarkeit

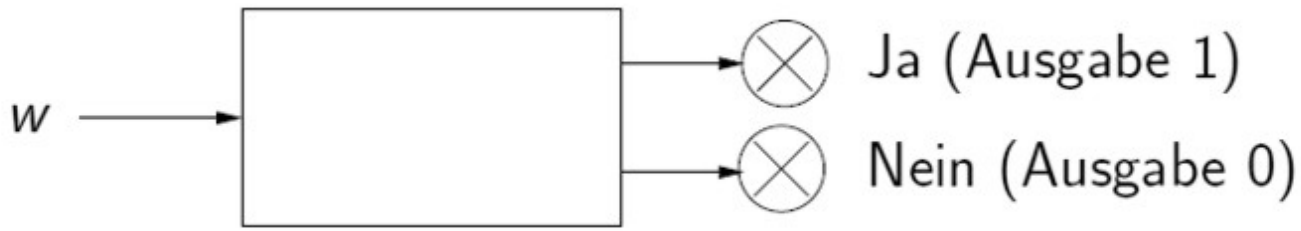
Entscheidungsproblem als Wortproblem von $L \subseteq \Sigma^*$:
Eingabe: $w \in \Sigma^*$
Aufgabe: Entscheiden, ob $w \in L$ gilt oder nicht.
Formalisierung (zur Erinnerung):

- Eine Sprache $L \subseteq \Sigma^*$ heißt entscheidbar, wenn es eine TM $M = (Q, \Sigma, \Gamma, \delta, z_0, E)$ gibt, die
  - auf jede Eingabe $w \in \Sigma^*$ terminiert
  - w *genau dann* akzeptiert, wenn $w \in L$ gilt.
    (M akzeptiert w falls: $z_0 w \vdash_M^* \alpha z_+ \beta$
    mit $\alpha, \beta \in \Gamma^*$, $z_+ \in E$ akzeptierender Zustand und $\alpha z_+ \beta$ finale Konfiguration.)
- **Alternativ:** Eine Sprache $L \subseteq \Sigma^*$ heißt entscheidbar, falls die totale charakteristische Funktion $\chi_L : \Sigma^* \to \{0, 1\}$ von L berechenbar ist:

$$\chi_L(w) = \begin{cases} 1, & \text{falls } w \in L \\ 0, & \text{sonst} \end{cases}$$

Entscheidbarkeit im Maschinenmodell

Veranschaulichung der Entscheidbarkeit im Maschinenmodell



bei jeder Eingabe rechnet die Maschine eine endliche Zeit und gibt dann „Ja“ oder „Nein“ aus

„Ausgeben“:

„Ja“	=	Halten in einem akzeptierenden Zustand
„Nein“	=	Halten in einem *nicht* akzeptierenden Zustand

Unentscheidbarkeit

Formalisierung (zur Erinnerung, Fortsetzung):

- Eine *unentscheidbare* Sprache ist eine Sprache, die nicht entscheidbar ist.

Anschauliche Interpretation:

- Eine Eigenschaft auf einer Menge heißt entscheidbar (auch: **rekursiv**), wenn es ein Entscheidungsverfahren für sie gibt.
- Ein Entscheidungsverfahren ist ein Algorithmus (oder eine Turing-Maschine), der (bzw. die) für jedes Element der Menge beantworten kann, ob es die Eigenschaft hat oder nicht.
- Wenn es ein solches Entscheidungsverfahren nicht gibt, dann nennt man die Eigenschaft unentscheidbar.

Entscheidungsprobleme (II)

Beispiel

Informelles Entscheidungsproblem (bzw. algorithmische Aufgabe):

(∗) „Entscheide (für beliebig gegebene n, m), ob die natürlichen Zahlen n und m teilerfremd sind.“

Formale Repräsentation: durch die Menge

$$\{(n, m) \in \mathbb{N}^2 \mid n \text{ und } m \text{ sind teilerfremd}\}$$

Kodierung: über dem Alphabet $\{|, \circ\}$ durch die formale Sprache

$$L_{tf} = \{|^n \circ |^m \in \{|, \circ\}^* \mid n, m \in \mathbb{N}; n \text{ und } m \text{ sind teilerfremd}\}$$

Entscheidungsprobleme (III)

Beispiel (Fortsetzung)

- L_{tf} ist im Sinne der formalen Definition entscheidbar
- TM, die L_{tf} entscheidet, kann verstanden werden als ein algorithmisches Verfahren zur Lösung von $(*)$
- **Informell:** das Entscheidungsproblem $(*)$ ist entscheidbar

  Algorithmus für $(*)$:
  Lies m, n;
  for $k := 2$ **to** $min(m, n)$ **do**
  if k teilt m und n **then**
  halte und gib „Nein“ aus
  endif
  enddo
  Gib „Ja“ aus.

  Das Prädikat „k teilt m und n“ ist total und berechenbar für $m, n > 1$ und alle $1 < k \leq min(m, n)$.

Programme als Eingabe von Programmen

- Algorithmen werden mithilfe von Programmen notiert
- Programme können als Wörter über einem geeigneten Alphabet kodiert werden
- Programme können dann selbst wieder Eingabe für einen Algorithmus sein
  (eventuell nach vorheriger Umkodierung als natürliche Zahl durch „Gödelisierung“)
- kommt in der Informatik häufiger vor

Beispiele

Compiler, Interpreter

Interessante Frage:

Kann ein Programm P_1, das ein anderes Programm P_2 als Eingabe erhält, entscheiden, ob das übergebene Programm P_2 eine bestimmte (nicht triviale) semantische Eigenschaft aufweist?

Das Halteproblem

- Berühmtes und oft betrachtetes Problem!
- **Allgemeines Halteproblem (H)**:
  Eingabe: Programm P, Eingabe x für P
  Aufgabe: Entscheiden, ob P auf Eingabe x nach endlich vielen Schritten terminiert.
- **Spezielles Halteproblem bzw. Selbstanwendungsproblem (SANW)**:
  Eingabe: Programm P
  Aufgabe: Entscheiden, ob P mit sich selbst als Eingabe nach endlich vielen Schritten terminiert.
- Für manche Programme leicht:
  z.B.: LOOP-Programme terminieren auf jede Eingabe
- **Frage:** Gibt es einen Algorithmus, der (Nicht-)Terminierung eines gegebenen Programms in endlicher Zeit feststellt?

Halteproblem für Turingmaschinen (I)

Formalisierung:

- Allgemeines Halteproblem für TM:
  Eingabe: Codewort $c(M)$ der (deterministischen) TM M, Eingabewort $w \in \Sigma^*$
  Aufgabe: Entscheiden, ob M auf Eingabe w terminiert
- Spezielles Halteproblem für TM:
  Eingabe: Codewort $c(M)$ der TM M
  Aufgabe: Entscheiden, ob M auf Eingabe $c(M)$ terminiert
- Also Wortprobleme der folgenden formalen Sprachen:

$$L_{\text{halt}} = \{c(M)w \mid M \text{ TM, die auf } w \text{ hält}\}$$
$$L_{\text{sanw}} = \{c(M) \mid M \text{ TM, die auf } c(M) \text{ hält}\}$$

Das spezielle Halteproblem (I)

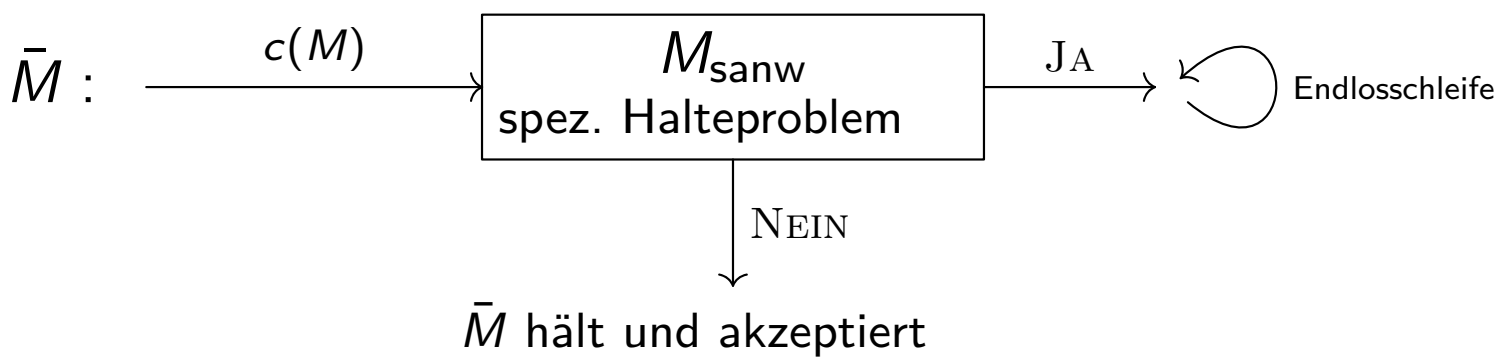
Satz 9.1

L_{sanw} ist unentscheidbar.

Beweis: durch Widerspruch.

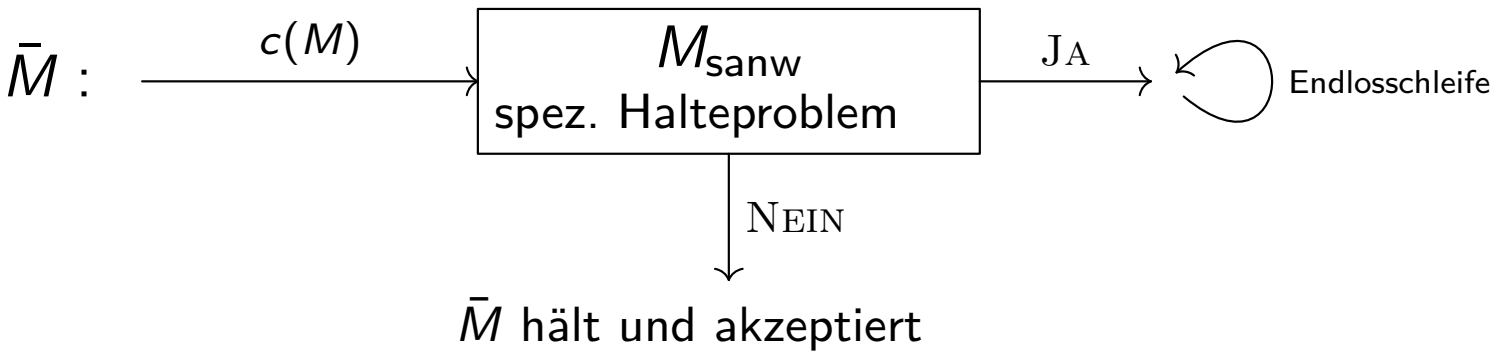
- Annahme: L_{sanw} ist entscheidbar.
- Dann existiert eine TM M_{sanw}, die L_{sanw} entscheidet.
- Mit M_{sanw} baue eine neue TM $\bar{M}$:

  $\bar{M}$ hält genau dann, wenn M_{sanw} hält und **nicht** akzeptiert
  (falls M_{sanw} hält und akzeptiert, hält $\bar{M}$ *nicht*)



Das spezielle Halteproblem (II)

Beweis (Fortsetzung):



Frage: Hält $\bar{M}$ bei Eingabe von $c(\bar{M})$ oder nicht?

- Falls **ja**: M_{sanw} hält auf $c(\bar{M})$ und akzeptiert

  $\rightsquigarrow$ $\bar{M}$ hält nicht auf $c(\bar{M})$. Widerspruch!
- Falls **nein**: M_{sanw} hält auf $c(\bar{M})$ und akzeptiert nicht

  $\rightsquigarrow$ $\bar{M}$ hält und akzeptiert $c(\bar{M})$. Widerspruch!

Fazit: Annahme falsch, d.h. L_{sanw} nicht entscheidbar.

Reduktionen (I)

Ziel jetzt: aus der Unentscheidbarkeit von L_{sanw}
weitere Unentscheidbarkeitsresultate gewinnen.

Dazu argumentiere:

1. Wenn man Problem L_2 lösen könnte, dann könnte man auch L_1 lösen (Reduktionsschritt).
2. Dann ist L_2 mindestens so schwierig wie L_1.
   (Notation: $L_1 \leq L_2$)
3. L_1 ist jedoch unentscheidbar.
4. Also: L_2 auch unentscheidbar.

Reduktionen (II)

Definition

Seien Σ ein Alphabet und $L_1, L_2 \subseteq \Sigma^*$ zwei Sprachen.

L_1 heißt **reduzierbar auf** L_2, wenn es eine totale, berechenbare Funktion $f : \Sigma^* \to \Sigma^*$ gibt, so dass für alle $w \in \Sigma^*$ gilt:

$$w \in L_1 \iff f(w) \in L_2.$$

(**Schreibweise:** $L_1 \leq L_2$ oder auch $f : L_1 \leq L_2$.)

Informell bedeutet $L_1 \leq L_2$:

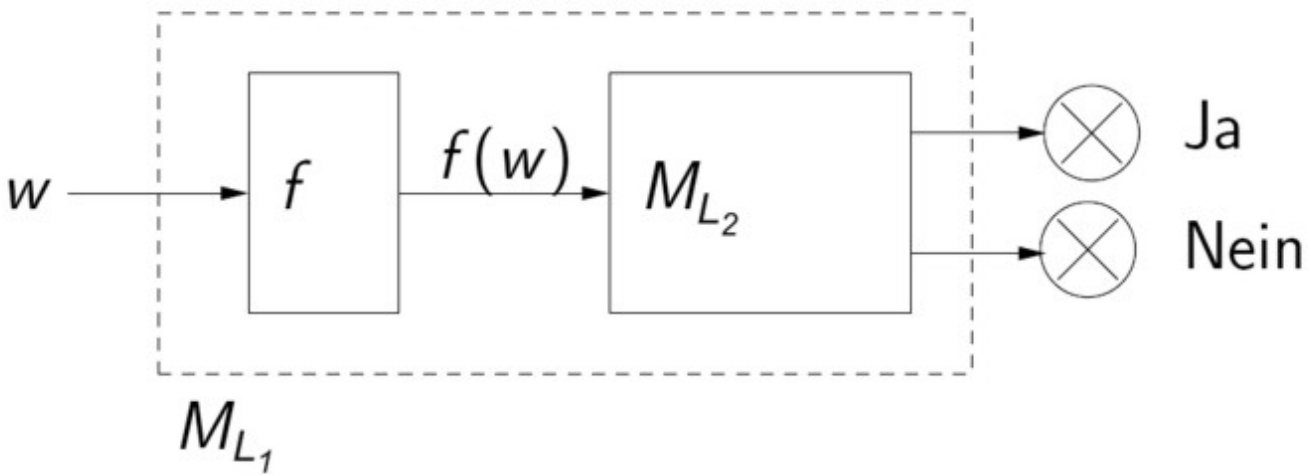
„L_2 mindestens so unentscheidbar ist wie L_1“.

Reduktionen (III)

Anschaulich: wenn $f : L_1 \leq L_2$, dann kann man aus

- einer TM M_{L_2}, die L_2 akzeptiert und
- einer TM, die f berechnet

eine TM M_{L_1} bauen, die L_1 akzeptiert:



Reduktionen (IV)

Satz 9.2

Seien Σ ein Alphabet, $L_1, L_2 \subseteq \Sigma^*$. Es sei $L_1 \leq L_2$.

- Ist L_2 entscheidbar, so ist auch L_1 entscheidbar.
- Ist L_1 unentscheidbar, so ist auch L_2 unentscheidbar.

Methode, um die Unentscheidbarkeit eines (als Sprache kodierten) Problems L zu zeigen:

- finde eine geeignete unentscheidbare Sprache L'
  (z.B. spezielles Halteproblem)
- finde eine geeignete Funktion f zur Reduktion von L' auf L und beweise (Korrektheit der Reduktion):
  1. f total und berechenbar
  2. $w \in L' \iff f(w) \in L$
- Dann folgt aus Satz 9.2: L ist unentscheidbar

Das allgemeine Halteproblem

Satz 9.3

Das allgemeine Halteproblem ist unentscheidbar.

Beweis: durch Reduktion des speziellen Halteproblems: $L_{\text{sanw}} \leq L_{\text{halt}}$

Definiere $f : \Sigma^* \to \Sigma^*$:

$$f(w) = \begin{cases} c(M)c(M) & \text{falls } w = c(M) \text{ für TM } M \\ 0 & \textit{sonst} \end{cases}$$

1. f ist total und berechenbar
   (Syntaxcheck „$w \stackrel{?}{=} c(M)$“; dann $c(M)$ kopieren)
2. $$\begin{aligned} w \in L_{\text{sanw}} &\iff w = c(M) \text{ für TM } M, \text{ die auf } c(M) \text{ hält} \\ &\iff f(w) = c(M)c(M) \text{ für TM } M, \text{ die auf } c(M) \text{ hält} \\ &\iff f(w) \in L_{\text{halt}}. \end{aligned}$$

Spezielles Wortproblem für eine fest gegebene Grammatik

Spezielles Wortproblem für die (fest vorgegebene) Grammatik $G = (V, \Sigma, P, S)$:

Eingabe: $w \in \Sigma^*$
Aufgabe: Entscheide, ob w von der Grammatik G erzeugt wird.

Satz 9.4

Das spezielle Wortproblem für eine Chomsky-Grammatik ist (im Allgemeinen) unentscheidbar.

Für spezielle (eingeschränkte) Grammatiken ist das Wortproblem aber entscheidbar. (z.B. für *kontextfreie* Grammatiken)

Allgemeines Wortproblem für Grammatiken

Folge: auch das **allgemeine Wortproblem für Grammatiken** ist unentscheidbar.

Eingabe: (Code einer) Grammatik G und Wort $w \in \Sigma^*$

Aufgabe: Entscheide, ob w von G erzeugt wird oder nicht.

Entscheidungsprobleme für kontextfreie Grammatiken

CYK-Algorithmus zeigt: das allgemeine Wortproblem für kontextfreie Grammatiken ist entscheidbar.

Einige unentscheidbare Probleme für kontextfreie Sprachen:

Satz 9.5

Folgende Probleme für gegebene kontextfreie Grammatiken G_1, G_2 sind unentscheidbar:

a) Schnittproblem: Ist $L(G_1) \cap L(G_2) = \emptyset$?

b) Äquivalenzproblem: Ist $L(G_1) = L(G_2)$?

c) Ist $L(G_1) = \Sigma^*$?
(Σ Terminalalphabet mit $|\Sigma| \geq 2$;
für $|\Sigma| = 1$ ist jedes kontextfreie L schon regulär)

Zum Vergleich:
für reguläre Sprachen sind diese Probleme entscheidbar!

Satz von Rice

- Jede nicht triviale Eigenschaft von berechenbaren Funktionen (oder formalen Sprachen) ist unentscheidbar.
- *Trivial* heißt eine Eigenschaft, wenn *alle oder keine* berechenbare Funktion (oder formale Sprache) die Eigenschaft hat.

Satz 9.6: Satz von Rice

Sei $\mathcal{R}$ die Klasse aller Turing-berechenbaren Funktionen und sei $\mathcal{S} \subseteq \mathcal{R}$ mit $\emptyset \neq \mathcal{S} \neq \mathcal{R}$.

Dann ist die folgende formale Sprache unentscheidbar:

$$L_{\mathcal{S}} = \{c(M) \mid \text{die von der TM } M \text{ berechnete Funktion liegt in } \mathcal{S}\}$$

Satz von Rice (Anwendungsbeispiel)

Beispiel

Betrachte das Problem:
Eingabe: $c(M)$ für TM M (die $f : \Sigma^* \to \Sigma^*$ berechnet)
Aufgabe: Entscheiden, ob $|f(w)|$ immer gerade Zahl ist oder nicht.

Als formale Sprache:

$$L_{\text{even}} = \{c(M) \mid M \text{ TM, die } f \text{ berechnet mit: } |f(w)| \text{ gerade für alle } w \in \Sigma^*\}$$

Dann gilt: $L_{\text{even}} = L_{\mathcal{S}}$ für

$$\mathcal{S} = \{f \mid f : \Sigma^* \to \Sigma^* \text{ berechenbar und } |f(w)| \text{ gerade } \forall w \in \Sigma^*\}$$

$\mathcal{S}$ nicht trivial (d.h. $\emptyset \neq \mathcal{S} \neq \mathcal{R}$), denn:

$f_1(w) = \varepsilon$ liegt in $\mathcal{S}$; $f_2(w) = 1$ liegt nicht in $\mathcal{S}$

$\Longrightarrow$ wegen Satz von Rice ist $L_{\text{even}} = L_{\mathcal{S}}$ unentscheidbar.

Satz von Rice (für formale Sprachen)

Satz 9.7: Satz von Rice (für formale Sprachen)

Sei $\mathcal{S}$ eine nicht leere, echte Teilmenge der Menge aller formalen Sprachen.

Dann ist die folgende formale Sprache unentscheidbar:

$$L_{\mathcal{S}} = \{c(M) \mid M \text{ ist TM und } L(M) \text{ liegt in } \mathcal{S}\}.$$

Beispiel

Die folgende Sprache ist unentscheibar:

$$L_{\emptyset} = \{c(M) \mid \text{die TM } M \text{ akzeptiert keine Eingabe}\}$$

Setze $\mathcal{S} = \{\emptyset\}$. Dann ist $\mathcal{S}$ „nicht triviale Eigenschaft“, denn:

$$\mathcal{S} \neq \emptyset \text{ und } \mathcal{S} \text{ enthält nicht alle Sprachen.}$$

Postsches Korrespondenzproblem (I)

- Das Alphabet Σ sei fest vorgegeben
- **Postsches Korrespondenzproblem (PCP):**
  Eingabe: endliche Folge von Paaren nicht leerer Wörter

$$(v_1, w_1), (v_2, w_2), \ldots, (v_k, w_k) \qquad v_i, w_i \in \Sigma^+, i = 1, \ldots, k$$

  Aufgabe: Entscheiden, ob es eine nicht-leere Folge $i_1, \ldots, i_n$ $(1 \leq i_j \leq k$ für $j = 1, \ldots, n)$ von Indizes gibt, so dass gilt:

$$v_{i_1} v_{i_2} \ldots v_{i_n} = w_{i_1} w_{i_2} \ldots w_{i_n}$$

- Die Folge $i_1, \ldots, i_n$ der Indizes heißt **Lösung** des Problems.

Postsches Korrespondenzproblem (II)

Beispiel 1

Gegeben seien die Paare $(1, 101), (10, 00), (011, 11)$ mit

$$\begin{array}{lll} v_1 = 1; & v_2 = 10; & v_3 = 011 \\ w_1 = 101; & w_2 = 00; & w_3 = 11 \end{array}$$

Das PCP besitzt in diesem Fall die Lösung $I_1 = (1, 3, 2, 3)$, denn:

$$\begin{aligned} v_1 \cdot v_3 \cdot v_2 \cdot v_3 &= 1 \cdot 011 \cdot 10 \cdot 011 \\ &= 101110011 \\ &= 101 \cdot 11 \cdot 00 \cdot 11 \\ &= w_1 \cdot w_3 \cdot w_2 \cdot w_3 \end{aligned}$$

Postsches Korrespondenzproblem (III)

Beispiel 2

Gegeben seien die Paare $(001, 0), (01, 011), (01, 101), (10, 001)$ mit

$$\begin{array}{llll} v_1 = 001; & v_2 = 01; & v_3 = 01; & v_4 = 10 \\ w_1 = 0; & w_2 = 011; & w_3 = 101; & w_4 = 001 \end{array}$$

Das PCP besitzt in diesem Fall die Lösung

$$I_1 = (2, 4, 3, 4, 4, 2, 1, 2, 4, 3, 4, 3, 4, 4, 3, 4, 4, 2, 1, 4, 4, 2, 1, 3, 4, 1, 1, 3, 4, 4, 4, 2, 1, 2, 1, 1, 1, 3, 4, 3, 4, 1, 2, 1, 4, 4, 2, 1, 4, 1, 1, 3, 4, 1, 1, 3, 1, 1, 3, 1, 2, 1, 4, 1, 1, 3).$$

Diese besteht schon aus 66 Paaren und zeigt die Komplexität des Problems.

Postsches Korrespondenzproblem (IV)

Satz 9.8

Das Postsche Korrespondenzproblem ist (im Allgemeinen) unentscheidbar.

Bemerkung:

- PCP wird ebenfalls als Ausgangsproblem für weitere Unentscheidbarkeitsaussagen benutzt
- „Reduktion“ von PCP auf Problem P

  $\Longrightarrow$ P auch unentscheidbar
- Zum Beispiel unentscheidbare Grammatikprobleme wie:
  Eingabe: Zwei (codes von) Chomsky-Grammatiken G_1 und G_2
  Aufgabe: Entscheiden ob $L(G_1) \cap L(G_2) = \emptyset$.

  (siehe z.B. Schöning)

Semi-entscheidbare Probleme (I)

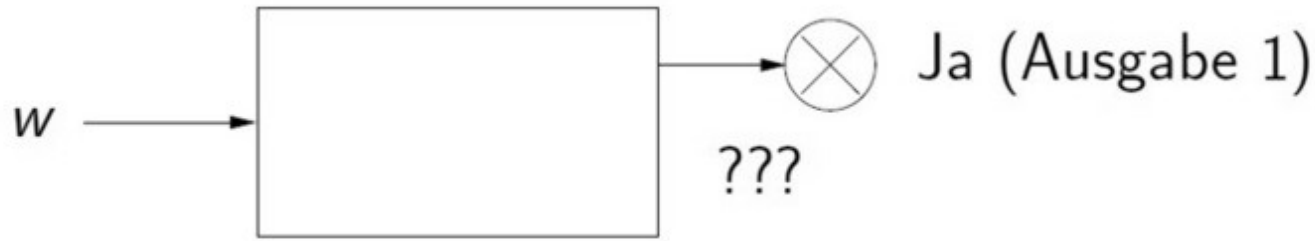
Erinnerung: Was heißt semi-entscheidbar?

- Semi-Entscheidbarkeit ist *schwächerer* Begriff als Begriff der Entscheidbarkeit
- Eine Sprache $L \subseteq \Sigma^*$ heißt semi-entscheidbar, wenn es eine TM M gibt, so dass $w \in L$ genau dann gilt, wenn M w akzeptiert.
- **Alternativ:** Eine Sprache $L \subseteq \Sigma^*$ heißt semi-entscheidbar, wenn die partielle charakteristische Funktion $\chi' : \Sigma^* \to \{0, 1\}$ von L berechenbar ist:

$$\chi'_L(w) = \begin{cases} 1, & \text{falls } w \in L \\ \text{undefiniert} & \text{sonst.} \end{cases}$$

Semi-Entscheidbarkeit im Maschinenmodell

Veranschaulichung der Semi-Entscheidbarkeit im Maschinenmodell



- bei jeder Eingabe rechnet die Maschine und gibt – falls $w \in L$ gilt – nach endlicher Zeit „ja“ aus
- falls die Antwort „nein“ lauten müsste, **kann** es sein, dass die Maschine nie terminiert (oder sie gibt nach endlicher Zeit „nein“aus)
- Folge: man kann sich nie sicher sein, ob die Maschine nicht doch irgendwann „ja“ ausgeben wird, da die Antwortzeit nicht beschränkt ist

Semi-entscheidbare Probleme (II)

- **Frage:** sind einige der zuvor betrachteten unentscheidbaren Probleme „wenigstens“ semi-entscheidbar?
- **Antwort:** Ja, das ist der Fall.

Semi-Entscheidbarkeit des speziellen Halteproblems

Satz 9.9

Das spezielle Halteproblem ist semi-entscheidbar.

Beweis: Benutze die universelle TM M_u wie folgt:

1. bei Eingabe von w, prüfe ob $w = c(M)$ für eine TM M (falls nicht, halte und akzeptiere w nicht; sonst weiter bei 2)
2. simuliere mit M_u die TM M auf Eingabe $c(M)$
3. halte und akzeptiere genau wenn M_u hält (und dabei akzeptiert oder nicht akzeptiert)

Analog:

Satz 9.10

Das allgemeine Halteproblem ist semi-entscheidbar.

Semi-Entscheidbarkeit des Wortproblems

Erinnerung: Satz 6.6

a) Zu jeder TM M gibt es eine Chomsky-Grammatik G mit $L(G) = L(M)$.

b) Zu jeder Chomsky-Grammatik G gibt es eine TM M mit $L(M) = L(G)$.

Daraus folgt die Semi-Entscheidbarkeit des speziellen Wortproblems für Grammatiken:

Satz 9.11

Ist G eine Chomsky-Grammatik, so ist $L(G)$ semi-entscheidbar.

Semi-entscheidbar und rekursiv aufzählbar (I)

Definition

Eine nicht-leere Sprache $L \subseteq \Sigma^*$ heißt rekursiv aufzählbar, falls $L = \emptyset$ oder falls es eine totale und berechenbare Funktion $f : \mathbb{N} \to \Sigma^*$ gibt, so dass

$$L = \{f(0), f(1), f(2), \dots\}$$

Sprechweise: „f zählt L auf.“
(Man beachte, dass $f(i) = f(j)$ zulässig ist.)

Satz 9.12

Eine Sprache ist rekursiv aufzählbar genau dann, wenn sie semi-entscheidbar ist.

Semi-entscheidbar und rekursiv aufzählbar (II)

Beweis von Satz 9.12:
(hier nur „$\Longrightarrow$“; andere Richtung „$\Longleftarrow$“ ohne Beweis)

- Sei L ist rekursiv aufzählbar mittels Funktion f.
- Semi-Entscheidungsverfahren:

input(x);
for $n := 0, 1, 2, 3, \ldots$ **do**
  if $f(n) = x$ **then output**(1) **endif**
endfor

Äquivalente Aussagen

Folgerung: Die folgenden Aussagen sind äquivalent für $L \subseteq \Sigma^*$:

- L ist semi-entscheidbar.
- L ist rekursiv aufzählbar.
- $L = L(M)$ für eine Turingmaschine M.
- χ'_L ist Turing-/WHILE-berechenbar.
- L ist Definitionsbereich einer berechenbaren Funktion.
- L ist Wertebereich einer berechenbaren Funktion.
- L ist vom Typ 0.

Nicht semi-entscheidbare Probleme

Erinnerung: Satz 6.3

Sei Σ ein Alphabet.

Es gibt Funktionen $f : \Sigma^* \to \Sigma^*$, die nicht berechenbar und Sprachen $L \subseteq \Sigma^*$, die nicht semi-entscheidbar (und damit auch nicht entscheidbar) sind.

Beispiel 1: Äquivalenzproblem für Turing-Maschinen

$$L_{\text{equiv}} = \{c(M_1)c(M_2) \mid M_i \ (i = 1, 2) \text{ TM mit } L(M_1) = L(M_2)\}$$

- Tatsächlich hat man im Bereich der unentscheidbaren Probleme eine ganze Hierarchie von (unendlich vielen) **Unentscheidbarkeitsgraden** mit „immer unentscheidbareren“ Problemen.
- Die (semi-)entscheidbaren Probleme bilden gerade den „untersten“ Grad dieser Hierarchie.

Ein nicht semi-entscheidbares Problem

Beispiel 2: Die Sprache L_{code}

Die folgende formale Sprache ist nicht semi-entscheidbar:

$$L_{\text{code}} = \{c(M) \mid M \text{ ist TM mit } c(M) \notin L(M)\}.$$

Beweis: durch Widerspruch.

- **Annahme:** L_{code} ist semi-entscheidbar.
- Also existiert eine TM $\bar{M}$ mit $L(\bar{M}) = L_{\text{code}}$.
- Setze $w := c(\bar{M})$. Es gilt $w \in L_{\text{code}}$ oder $w \notin L_{\text{code}}$.
- **1. Fall**: $w \in L_{\text{code}}$
  Dann gilt wegen Def. von L_{code}: $w \notin L(\bar{M}) = L_{\text{code}}$. Widerspruch!
- **2. Fall**: $w \notin L_{\text{code}}$
  Dann gilt: $w \in L(\bar{M}) = L_{\text{code}}$. Widerspruch!
- **Also:** Annahme ist falsch! Somit ist L_{code} nicht semi-entscheidbar.

Zusammenfassung