6.4 Entscheidbarkeit

	Wortproblem	Leerheit	Äquivalenz	Schnittproblem
Typ 3	ja	ja	ja	ja
DCFL	ja	ja	ja	nein (*)
Typ 2	ja	ja	nein (*)	nein
Typ 1	ja	nein (*)	nein	nein
Typ 0	nein (*)	nein	nein	nein

(*) Diese Ergebnisse werden im Abschnitt 3 des nächsten Kapitels gezeigt. Dass in jeder Spalte die darunterliegenden Einträge dann ebenfalls „nein“ sein müssen, ist klar.

Kapitel II Berechenbarkeit, Entscheidbarkeit

1. Der Begriff der Berechenbarkeit

Unsere Vorstellung ist:
$f : \mathbb{N}_0^k \to \mathbb{N}_0$ ist berechenbar, wenn es einen Algorithmus gibt, der f berechnet, bzw, genauer, der bei jeder Eingabe $(n_1, \ldots, n_k) \in \mathbb{N}_0^k$ nach endlich vielen Schritten mit dem Ergebnis $f(n_1, \ldots, n_k) \in \mathbb{N}_0$ hält.

Was bedeutet „Algorithmus“ an dieser Stelle?

AWK, B, C, Euler, Fortran, Haskell, Id, JAVA, Lisp, Modula, Oberon, Pascal, Simula, . . .-Programme?
Gibt es einen Unterschied, wenn man sich auf eine bestimmte Programmiersprache beschränkt?

Analog für partielle Funktionen

$$f : \mathbb{N}_0^k \supseteq D \to \mathbb{N}_0$$

bedeutet berechenbar Folgendes:

1. Algorithmus hält nach endlich vielen Schritten mit dem richtigen Ergebnis, wenn $(n_1, \ldots, n_k) \in D$;
2. hält nicht, wenn $(n_1, \ldots, n_k) \notin D$.

Beispiel 119

Wir definieren folgende Funktionen:

$$f_1(n) = \begin{cases} 1 & \text{falls } n \text{ als Ziffernfolge Anfangsstück von } \pi \text{ ist} \\ 0 & \text{sonst} \end{cases}$$

$$f_2(n) = \begin{cases} 1 & \text{falls } n \text{ interpretiert als Ziffernfolge in } \pi \text{ vorkommt} \\ 0 & \text{sonst} \end{cases}$$

$$f_3(n) = \begin{cases} 1 & \text{falls mindestens } n \text{ aufeinanderfolgende Ziffern} \\ & \text{in } \pi \text{ gleich 1 sind} \\ 0 & \text{sonst} \end{cases}$$

$\pi = 3,14159265358979323846264338327950288419716939937\ldots$

Einige Beispiele sind damit:

$$f_1(314) = 1,\, f_1(415) = 0,\, f_2(415) = 1\ .$$

Beispiel 119

$$f_1(n) = \begin{cases} 1 & \text{falls } n \text{ als Ziffernfolge Anfangsstück von } \pi \text{ ist} \\ 0 & \text{sonst} \end{cases}$$

Wie man leicht einsieht, ist f_1 berechenbar, denn um festzustellen, ob eine Ziffernfolge ein Anfangsstück von π ist, muss π nur auf entsprechend viele Dezimalstellen berechnet werden.

Beispiel 119

$$f_2(n) = \begin{cases} 1 & \text{falls } n \text{ interpretiert als Ziffernfolge in } \pi \text{ vorkommt} \\ 0 & \text{sonst} \end{cases}$$

Für f_2 wissen wir nicht, ob es berechenbar ist. Um festzustellen, dass die Ziffernfolge in π vorkommt, müsste man π schrittweise immer genauer approximieren. Der Algorithmus würde stoppen, wenn die Ziffernfolge gefunden wird. Aber was ist, wenn die Ziffernfolge in π nicht vorkommt?

Vielleicht gibt es aber einen (noch zu findenden) mathematischen Satz, der genaue Aussagen über die in π vorkommenden Ziffernfolgen macht.

Wäre π vollkommen zufällig, was es aber nicht ist, dann würde jedes n als Ziffernfolge irgendwann vorkommen.

Beispiel 119

$$f_3(n) = \begin{cases} 1 & \text{falls mindestens } n \text{ aufeinanderfolgende Ziffern} \\ & \text{in } \pi \text{ gleich 1 sind} \\ 0 & \text{sonst} \end{cases}$$

f_3 ist berechenbar, denn $f_3 \equiv f_4$, mit

$$f_4(n) = \begin{cases} 1 & n < n_0 \\ 0 & \text{sonst} \end{cases}$$

wobei $n_0 \in \mathbb{N} \cup \{\infty\}$ die maximale Anzahl von aufeinanderfolgenden 1en in π ist. Hierbei ist es nicht von Bedeutung, wie die Zahl n_0 berechnet werden kann - wichtig ist nur, dass eine solche Zahl $n_0 \in \mathbb{N} \cup \{\infty\}$ existiert.

Hinweis:
Viele interessante Informationen zur Kreiszahl π findet man z.B. bei

- Wikipedia
- Yasumasa Kanada's Lab; diese Webseite enthält auch Angaben zur Verteilung der Ziffern in der (Dezimal-/Hexadezimal)-Bruchdarstellung von π.

Weitere Vorschläge, den Begriff der Berechenbarkeit zu präzisieren und zu formalisieren:

1. Turing-Berechenbarkeit
2. Markov-Algorithmen
3. λ-Kalkül
4. μ-rekursive Funktionen
5. Registermaschinen
6. AWK, B, C, Euler, Fortran, Id, JAVA, Lisp, Modula, Oberon, Pascal, Simula, ...-Programme
7. while-Programme
8. goto-Programme
9. DNA-Algorithmen
10. Quantenalgorithmen
11. u.v.a.m.

Es wurde bewiesen: Alle diese Beschreibungsmethoden sind in ihrer Mächtigkeit äquivalent.

Church'sche These

Dieser formale Begriff der Berechenbarkeit stimmt mit dem intuitiven überein.

Definition 120

Eine (partielle) Funktion

$$f : \mathbb{N}_0^k \supseteq D \to \mathbb{N}_0$$

heißt Turing-berechenbar, falls es eine deterministische Turingmaschine gibt, die für jede Eingabe $(n_1, \ldots, n_k) \in D$ nach endlich vielen Schritten mit dem Bandinhalt

$$f(n_1, \ldots, n_k) \in \mathbb{N}_0$$

hält. Falls $(n_1, \ldots, n_k) \notin D$, hält die Turingmaschine nicht!

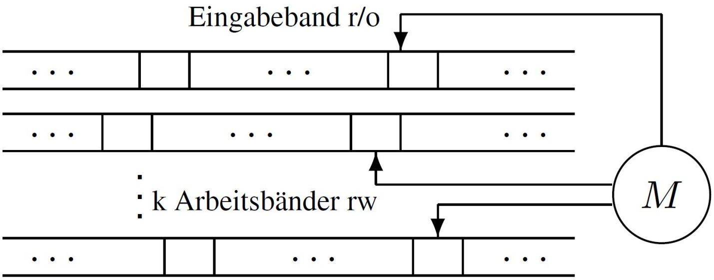
Dabei nehmen wir an, dass Tupel wie $(n_1, \ldots, n_k)$ geeignet codiert auf dem Band der Turingmaschine dargestellt werden.

Eine beliebte Modellvariante ist die k-Band-Turingmaschine, die statt einem Band k, $k \geq 1$, Arbeitsbänder zur Verfügung hat, deren Lese-/Schreibköpfe sie unabhängig voneinander bewegen kann.

Oft existiert auch ein spezielles Eingabeband, das nur gelesen, aber nicht geschrieben werden kann (read-only). Der Lesekopf kann jedoch normalerweise in beiden Richtungen bewegt werden.

Ebenso wird oft ein spezielles Ausgabeband verwendet, das nur geschrieben, aber nicht gelesen werden kann (write-only). Der Schreibkopf kann dabei nur nach rechts bewegt werden.

Beispiel 121 (k-Band-Turingmaschine)



Satz 122

Jede k-Band-Turingmaschine kann effektiv durch eine 1-Band-TM simuliert werden.

Beweis:

Wir simulieren die k Bänder auf k Spuren eines Bandes, wobei wir das Teilalphabet für jede Spur auch noch so erweitern, dass die Kopfposition des simulierten Bandes mit gespeichert werden kann.

Definition 123

Eine Sprache $A \subseteq \Sigma^*$ heißt rekursiv oder entscheidbar, falls es eine deterministische TM M gibt, die auf allen Eingaben $\in \Sigma^*$ hält und A erkennt.

Definition 124

Eine Sprache $A \subseteq \Sigma^*$ heißt rekursiv aufzählbar (r.a.) oder semi-entscheidbar, falls es eine TM N gibt, für die

$$L(N) = A\,,$$

also, falls A Chomsky-0 ist.

Definition 125

Sei $A \subseteq \Sigma^*$. Die charakteristische Funktion χ_A von A ist

$$\chi_A(w) = \begin{cases} 1 & \text{falls } w \in A \\ 0 & \text{sonst} \end{cases}$$

Definition 126

Sei $A \subseteq \Sigma^*$. χ'_A ist definiert durch

$$\chi'_A(w) = \begin{cases} 1 & \text{falls } w \in A \\ \text{undefiniert} & \text{sonst} \end{cases}$$

Satz 127

A ist rekursiv $\Leftrightarrow$ χ_A ist berechenbar.

Beweis:

Folgt aus der Definition von rekursiv: es gibt eine TM, die `ja` oder `nein` liefert. Wandle das Ergebnis in 1 oder 0.

Satz 128

A ist rekursiv aufzählbar $\Leftrightarrow$ χ'_A ist berechenbar.

Beweis:

Folgt unmittelbar aus der Definition.

Satz 129

A ist rekursiv $\Leftrightarrow$ χ'_A und $\chi'_{\bar{A}}$ sind berechenbar ($\Leftrightarrow$ A und $\bar{A}$ sind r.a.)

Beweis:

Nur $\Leftarrow$ ist nichttrivial. Wir lassen hier eine TM für A und eine TM für $\bar{A}$ Schritt für Schritt parallel laufen.