Prof. Dr. Rolf Wanka
Bernd Bassimir, Alexander Raß, Jonas Schreiner,
Stefan Schwanecke, Jeanette Wohnig

Erlangen, 20. November 2018

Abgabe $\boxed{\text{Di., 27.11.18, 23:59 Uhr}}$

Übungen zur Vorlesung

Berechenbarkeit und Formale Sprachen

WS 2018/2019

Blatt 4

Je mehr Plus-Zeichen $+$ bei einer Aufgabe, desto wichtiger, je mehr Sterne $\star$, desto schwieriger.

AUFGABE 20:
[Präsenzaufgabe, $++++,\star\star\star$] Zeigen Sie:

$$L = \{\langle M\rangle w \mid M \text{ ist deterministische 1-Band-TM, die genau für die Eingabe } w \text{ hält}\}$$

ist nicht **rekursiv aufzählbar**.

Hinweis: Zum Beweis können Sie benutzen, daß in der Vorlesung gezeigt wurde, daß das Komplement $\overline{H}_\varepsilon = \{0,1\}^* \setminus H_\varepsilon$ des initialen Halteproblems nicht rekursiv aufzählbar ist. Mit anderen Worten: Zeigen Sie $\overline{H}_\varepsilon \leq L$. Dazu müssen Sie sich eine andere Stelle als üblich suchen, an der Sie das „starte M mit leerem Band“ in $\langle \text{feste\_Maschine}_{\langle M\rangle}\rangle$ einsetzen.

AUFGABE 21 (4 Punkte)**:**
[$+++++,\star\star$] M bezeichne eine deteterministische 1-Band-Turingmaschine. Betrachten Sie die Sprache

$$L = \{\langle M\rangle \mid M \text{ hält mindestens für die Eingabe } 0101\} \quad .$$

Zeigen Sie:

(a) L ist unentscheidbar. Führen Sie eine Reduktion(!) durch.

(b) L ist rekursiv aufzählbar. (Gewinnen Sie ein Gefühl für den Unterschied zwischen hier „mindestens“ und „genau“ in der Formulierung in Aufgabe 20.)

AUFGABE 22 (4 Punkte)**:**
[$+++,\star\star$] M bezeichnet im folgenden immer eine deteterministische 1-Band-Turingmaschine. Zeigen Sie mittels Reduktion, daß die folgenden Sprachen unentscheidbar sind.

(a) $L_1 := \{\langle M\rangle \mid M \text{ hält für jede Eingabe}\}$ (Totalitätsproblem)

(b) $L_2 := \{\langle M_1\rangle\langle M_2\rangle \mid L(M_1) = L(M_2)\}$ (Äquivalenzproblem)

AUFGABE 23 (4 Punkte)**:**
[$+++,\star$] Gegeben sei die folgende rekursiv definierte Funktion $u : \mathbb{N}^{\neq 0} \to \mathbb{N}^{\neq 0}$:

$$u(n) = \begin{cases} 1 & \text{für } n = 1 \\ u(n/2) & n \text{ gerade} \\ u(3 \cdot n + 1) & \text{sonst} \end{cases}$$

u ist die Ulamsche Funktion und kann nur den Wert 1 annehmen.

(a) Führen Sie die Berechnung von $u(15)$, $u(31)$, $u(409\,599)$ und $u(409\,600)$ aus und bestimmen Sie die größten Werte x, für die $u(x)$ dabei unterwegs jeweils aufgerufen wird (inklusive dem Startwert).

(b) Was entdecken Sie, wenn Sie den Definitionsbereich erweitern und $u(-5)$ berechnen wollen?

Hintergrund: Bis heute ist nicht bekannt, ob $u(n)$ für **alle** $n \in \mathbb{N}$ definiert ist. Die *Ulamsche Vermutung* ist, daß $u(n) = 1$ für alle $n \in \mathbb{N}^{\neq 0}$ ist.

Wenn Sie nun ein Programm $\langle M \rangle$ schreiben, das $u(n)$ berechnet (vermutlich haben Sie es ja zur Lösung der Aufgabe gemacht ☺), wäre die Ulamsche Vermutung bewiesen, wenn die in Aufgabe 22 vorgestellten Probleme entscheidbar wären, denn dann könnte man z. B. einfach fragen, ob $\langle M \rangle \in L_1$ ist.

AUFGABE 24:
[4 **Bonus**-Punkte, Abgabe nur dieser Aufgabe bis **4. Dezember 2018 in der Vorlesung**]
(*Hinweis: Es kommt zwar viel Text, aber ihn zu lesen lohnt sich in vielerlei Hinsicht.*)

Wir sagen im folgenden: Eine Turingmaschine berechnet eine Funktion $f : \mathbb{N} \to \mathbb{N}$, falls sie für alle $n \in \mathbb{N}$ bei Eingabe 1^{n+1} nach endlich vielen Schritten hält, und dann auf dem Band $1^{f(n)+1}$ steht. Eine solche Funktion wird im Englischen als *total recursive* bezeichnet.
Der folgende Text ist abgeschrieben aus dem Buch: J. BARWISE (Editor), "Handbook of Mathematical Logic", North-Holland Publishing Company, Amsterdam, 1977. Führen Sie den dort bloß skizzierten Beweis aus.

> We can give a specific example of a non-computable function by using the "busy beaver competition" of RADO [1962]. An n-state entry in this competition is a Turing machine M with $\Sigma = \{1\}$, $\Gamma = \{B, 1\}$ and $n+1$ states $q_0, \ldots, q_n$, the last of which is used only for halting (both $\delta(q_n, B)$ and $\delta(q_n, 1)$ are undefined), and such that when started on a blank tape, M eventually halts. When M does halt, its score in the competition is the number of 1's on the tape. Thus the machine tries to write as many 1's on the tape as it possibly can, but it must halt. Let $\mathrm{BB}(n)$ be the maximum possible score for an n-state entry.
>
> 2.2 THEOREM (RADO [1962]). *The function* BB *ist not computable. In fact for any total recursive f on $\mathbb{N}$, we have $f(x) < \mathrm{BB}(x)$ for all sufficiently large x.*
>
> PROOF (SKETCH). The function whose value at x is
>
> $$\max[f(2x+2), f(2x+3)]$$
>
> is computable, and hence is computed by some machine M having, say, k states. For each x, consider a machine N_x that writes 1^{x+1} on a blank tape and then behaves like M. Then N_x is a $(x+k+2)$-state entry in the busy beaver competition. So its score (the number displayed above plus 1) is bounded by $\mathrm{BB}(x+k+2)$, which for all $x \geq k$ is bounded by $\mathrm{BB}(2x+2)$. □
>
> T. RADO, *On non-computable functions*, The Bell System Technical Journal, vol. 41, no. 3, pp. 877–884, May 1962

Es reicht natürlich nicht, diesen Text einfach zu übersetzen. Insbesondere ist noch nicht klar, warum nun BB nicht berechenbar ist.
Zum Verständnis ganz wichtig: Schauen Sie zur Sicherheit das Wort „eventually" in einem Wörterbuch nach, es hat eine andere Bedeutung als das deutsche Wort „eventuell"! Und das Wort „Sketch" ist im Englischen weniger lustig als im Deutschen.

Übrigens, bis heute sind die folgenden Werte für BB bekannt:

n	1	2	3	4	5	6	$\cdots$
$\mathrm{BB}(n)$	1	4	6	13	≥ 4098	$\geq 3{,}514 \cdot 10^{18267}$	$\cdots$

Wir haben inzwischen ziemlich oft über „Turing hat 1936 gezeigt“ gesprochen. Deswegen ist vielleicht jetzt ein guter Zeitpunkt, daß Sie einmal der Neugier wegen in den Original-Aufsatz von Alan Turing hineinschauen. Sie finden den Aufsatz

> A. M. Turing. On Computable Numbers, With an Application to the Entscheidungsproblem, *Proc. London Math. Soc.*, 2(42) (1936), 230–265.

im pdf-Format wiedergegeben unter den URLs (zum Anklicken auch auf der Homepage unserer Vorlesung)

`http://plms.oxfordjournals.org/content/s2-42/1/230.full.pdf` oder
`http://www.thocp.net/biographies/papers/turing_oncomputablenumbers_1936.pdf`

Im Wissenschaftsjargon werden Aufsätze heutzutage als *Papers* bezeichnet.

> Dieses Paper ist erstaunlich einfach zu lesen. Natürlich brauchen Sie etwas Vertrautheit mit dem Wissenschaftsenglisch, und Sie müssen Turings Begriffe mit den Begriffen in Übereinstimmung bringen, die Sie in unserer Vorlesung kennenlernen. An dem Wissenschaftsenglisch führt für Sie kaum ein Weg vorbei, da im weiteren Verlauf des Studiums der überwältigende Teil der Fachliteratur in englischer Sprache sein wird. Insbesondere sind praktisch alle Papers, die in der Regel den Stand der Wissenschaft beschreiben, auf Englisch verfaßt. Sehen Sie sich mal auf den Webseiten der diversen Lehrstühle unter „Veröffentlichungen“, „Publikationen“ usw. um.
>
> Das deutsche Wort „Entscheidungsproblem“ im Titel ist der feststehende Begriff für das „Entscheidungsproblem in der Prädikatenlogik“, das David Hilbert 1928 im Nachgang zu seinen berühmten, bereits von 10 auf 23 angewachsenen Problemen — insbesondere zum 2. Problem, das dann Kurt Gödel 1930 gelöst hat — aus dem Jahr 1900 formuliert hat. In der Zeit bis vor dem zweiten Weltkrieg wurden mathematische Aufsätze überwiegend auf Deutsch verfaßt.

Turings Paper nun enthält in §1 und §2 die Definition des Rechner-Modells „Computing Machine“. Wir sagen heute dazu Turingmaschine☺. In §2 wird unter der Bezeichnung „computable“ das eingeführt, was wir „rekursiv aufzählbar“ nennen. Damit wird für uns auch der Anfang des Titels des Papers klar. Gemeint ist also: Über rekursiv aufzählbare Sprachen.

Wenn Sie das Paper weiter lesen, finden Sie die Beschreibung von universellen Turingmaschinen und von Gödelnummern (als „Description Number of the machine“, kurz: D.N), und, in §8, die Unentscheidbarkeit des Halteproblems, formuliert als “… whether a given number is the D.N (für uns: Gödelnummer) of a circle-free (für uns: mit der Eingabe haltenden) machine, and we have no general process for doing this in a finite number of steps. In fact, by applying the diagonal process argument correctly, we can show that there cannot be such general process.”

§9 ist schließlich die ausführliche Argumentation für die Church-Turing-These, die bereits in §1 vorweggenommen wurde: “It is my contention (hier in der Bedeutung: Behauptung, These) that these operations (einer TM) include all those which are used in the computation of a number.”

Dies alles erstaunt umso mehr, als Turing natürlich keine Computer kannte, wie wir sie jeden Tag benutzen. Zumindest war er aber an der Entwicklung der ersten britischen Computer Colossus, Bombe und ACE maßgeblich beteiligt.

Durch ein Paper aus dem Jahr 1950 mit dem Titel „Computing machinery and intelligence“ hat Turing schließlich auch noch die KI begründet. In diesem Aufsatz schlägt er den heute als Turing-Test bezeichneten Test vor. Das berühmte Programm ELIZA von Joseph Weizenbaum ist vor diesem Hintergrund zu sehen: Kann ein Mensch entscheiden, ob er mit einem Computer-Programm spricht? Wer den Film *Der Blade-Runner* gesehen hat, erkennt den Einfluß dieser Forschung bis in die moderne populäre Kunst.