Church und das Entscheidungsproblem

Warum wir noch Mathematiker brauchen

Joachim Breitner*

24. Juni 2011
Gulaschprogrammiernacht 11†, Karlsruhe

Zusammenfassung

Hilbert hatte einen Traum: Er wünschte sich eine formale Sprache, in der alle Mathematik formuliert werden kann, die widerspruchsfrei und vollständig ist und eine Maschine, die jede Aussage als wahr oder falsch identifiziert. Dann kam Church und weckte Hilbert aus seinem Traum: So eine Maschine kann es nicht geben, wir brauchen also weiter Mathematiker um Beweise zu finden.

Nach einer logikgeschichtlichen Einordnung werden wir sehen wie Church sein junges Lambda-Kalkül, die Mutter aller funktionalen Programmiersprachen, verwendet um das Entscheidungsproblem negativ zu lösen – wenige Monate bevor dies auch von Turing mit seiner bekannteren Arbeit zum Halteproblem gezeigt wurde.

Herzlich willkommen zum vielleicht theoretischsten Vortrag der diesjährigen Gulaschprogrammiernacht. Ich werde euch erst einen kurzen Einblick in den Stand der Mathematik und der Logik am Anfang des zwanzigsten Jahrhundert geben und dann zeigen, wie Church seinen Lambda-Kalkül nutzte um zu zeigen dass Hilberts Entscheidungsproblem nicht lösbar ist. Ich gehe jederzeit gerne auf Fragen ein – lieber komme ich nicht durch als dass ich mein Publikum zwischendurch verliere.

1 Das frühe zwanzigste Jahrhundert

Das früher zwanzigste Jahrhundert muss eine spannende Zeit für Mathematiker gewesen sein. Man hatte erkannt, dass man eine formale Sprache für mathematische Aussagen und

* mail@joachim-breitner.de
† http://entropia.de/GPN11

Beweise braucht, aber es war erstaunlich schwer, einen guten Formalismus zu finden. Schon im neunzehnten Jahrhundert hat George Boole die Aussagenlogik formalisiert. Gottlob Frege veröffentlichte später ein vielbeachtetes Werk, genannt „Begriffsschrift“, in der die Prädikatenlogik mit einer seltsamen zwei-dimensionalen Syntax formalisierte. Zwanzig Jahre später, das zwanzigste Jahrhundert hat gerade begonnen, schrieb Bertrand Russell einen Brief an Frege und wies hin auf das hin, was inzwischen als Russell's Paradox bekannt ist: Wir können eine Menge R definieren als

$$s \in R \iff s \notin s$$

und dann ersetzten wir s durch R und stellen fest

$$R \in R \iff R \notin R$$

was offensichtlich ein Problem ist. Russell zeigte dass das System der Begriffsschrift nicht konsistent ist.

Später startete Russell zusammen mit Alfred Whitehead die „principia mathematica“, ein mehrbändiges Werk dass die gesamte bekannte Mathematik auf eine solide formale Basis stellte. Sie wussten nun um das Problem des Selbst-Bezugs und vermieden Paradoxa mit Typtheorie: Man trennte strikt zwischen Individuen der Logik, Mengen von Individuen, Mengen von Mengen von Individuen und so weiter. An dieser Stelle kann man gut Eindruck schinden indem man sagt dass die Aussage 1+1=2 erst auf Seite 379 bewiesen wird, aber ich verzichte darauf. Alles in allem war es eine sehr mühsamer, aber wertvoller und respektierter Ansatz.

Nun betritt David Hilbert unsere Bühne, und aus irgend einem Grund erscheint er immer nur als Fragesteller, nie als Antwortgeber. Ich schließe mal dass wenn du einer der größten Mathematiker aller Zeiten werden willst, verschwende deine Zeit nicht mit Antworten… Jedenfalls, Hilberts Programm verlangte nach einem formalen System für alle Mathematik, in dem alle wahren Aussagen bewiesen werden können, keine Widersprüche vorkommen und für das es ein Entscheidungsverfahren gibt, dass zu jeder Aussage herausfindet, ob es wahr oder falsch ist.

Das sind ganz schön viele Wünsche auf einmal, und leider ist diese Welt nicht gut genug. Der erste Rückschlag kam 1932 von Kurt Gödel. Seinem ersten Unvollständigkeitssatz nach gibt es in jeder widerspruchsfreien Logik, die mächtig genug ist um über einfach Arithmetik zu reden, Formeln die wahr sind, aber nicht im System als wahr bewiesen werden. Kurz gesagt: Man kann nicht sowohl Widerspruchsfreiheit als auch Vollständigkeit haben.

Das war verständlicherweise eine Enttäuschung für Hilbert und ein Schock für die Mathematik an sich. Aber was ist mit Hilberts drittem Wunsch: Können wir wenigstens ein Entscheidungsverfahren haben, also eine mechanische Methode oder, wie wir heute sagen würden, ein Algorithmus der entscheidet ob eine Aussage beweisbar ist? Diese Frage heißt das Entscheidungsproblem und, armer Hilbert, auch das ist nicht möglich. Dies wurde zuerst von Alonzo Church 1935 mit seinem Lambda-Kalkül gezeigt, mittels eines schönen Widerspruchsbeweis.

Turing, der damals an Turingmaschinen gearbeitet hat, auch wenn er sie natürlich nicht so nannte, veröffentlichte sein Ergebnis zum Halteproblem, was auch die Unlösbarkeit des Entscheidungsproblems implizierte, nur wenige Monate nach Church. Dass mehr Leute von Turings Arbeit wissen liegt wohl daran dass mehr Informatikstudenten sich mit Turing-Maschinen als mit Lambdakalkül beschäftigen müssen.

Beide Ergebnisse, das von Church und das von Turing, haben allerdings ein Problem: Sie basieren auf einer unbewiesenen Hypothese! Hilbert fragte nach einer mechanischen Methode um die Entscheidbarkeit einer Formel zu beweisen, etwas das man auch „effektiv berechenbar“ nannte. Er hatte natürlich im Kopf dass ein mathematisch gebildeter Mensch streng eine Reihe von Regeln abarbeitet. Aber Mathematiker, und Menschen im Allgemeinen, wurden glücklicherweise noch nicht formalisiert. Daher kann Church und Turing gar nicht über eine solche mechanische Methode reden. Stattdessen schlagen sie ein formales Modell für Berechenbarkeit vor – das Lambdakalkül beziehungsweise Turingmaschinen – und argumentieren dann dass alles, was effektiv berechenbar ist, sich auch damit berechnen ließe. Um dies zu unterstreichen zeigte Church dass das Lambdakalkül genau so mächtig ist wie sogenannte „allgemein-rekursive Funktionen“, etwas das Gödel definiert und damals von vielen Untersucht wurde, und flehte dann in seinem Paper:

> “If this interpretation or some similar one is not allowed, it is difficult to see how the notion of an algorithm can be given any exact meaning at all.”

Aber ob sein Lambda-Kalkül wirklich alles abdeckt, was effektiv berechnen kann, wurde von einigen, einschließlich Gödel bezweifelt. Turing zeigte dann dass Turingmaschinen und das Lambdakalkül gleich mächtig sind und seine Argumentation – wohlgemerkt kein Beweis – dass diese Maschinen ein gutes Modell für Berechenbarkeit sind hat zumindest Gödel überzeugt und seit dem spricht man von der Church-Turing-These – aber natürlich ist das keine echte These, da es nie bewiesen wurde und nicht bewiesen werden kann.

2 Das Lambda-Kalkül

Nach diesem kleinen historischen Exkurs werfen wir nun einen Blick auf das Lambda-Kalkül. Das Lambda-Kalkül macht Aussagen über Lambda-Ausdrücke, die ziemlich einfach aufgebaut sind. Sei V eine Menge von Variablen, dann ist jede Variable schon ein Lambda-Ausdruck. Weiter ist die Anwendung oder Applikation eines Lambda-Ausdrucks auf einen anderen wieder ein Lambda-Ausdruck. Und zuletzt ist die Lambda-Abstraktion einer freien Variable in einem Lambda-Ausdruck wieder ein Lambda-Ausdruck. Eine Variable hießt frei wenn sie im Ausdruck vorkommt, aber nicht schon durch eine Lambda-Abstraktion gebunden ist.

$$
\begin{aligned}
v \in V &\implies v \in \Lambda \\
M, N \in \Lambda &\implies MN \in \Lambda \\
M \in \Lambda,\ x \in V,\ x \text{ frei in } M &\implies \lambda x.M \in \Lambda
\end{aligned}
$$

Hier haben wir ein paar Beispiele für Lambda-Ausdrücke, mit willkürlich gewählten Namen:

$$I = (\lambda x.x)$$
$$\omega = (\lambda x.xx)$$
$$C_3 = (\lambda f.(\lambda x.f(f(fx))))$$

Und was kann man nun mit diesen Ausdrücken machen? Nicht wirklich viel, es gibt eine Regel, die sogenannte Beta-Reduktion, die wir auf einen Lambda-Ausdruck oder einen seiner Unterterme anwenden können:

$$(\lambda x.M)N \longrightarrow M[x := N]$$

Diese Regel ersetzt eine Anwendung einer Lambda-Abstraktion durch den abstrahierten Ausdruck, wobei die abstrahierte Variable durch den zweiten Ausdruck in der Anwendung ersetzt wird. Wenn wir diese Regel wiederholt auf den Ausdruck M anwenden und so den Ausdruck N erhalten, sagen wir dass M zu N reduziert werden kann und schreiben mit Sternchen $M \longrightarrow^* N$.

Wichtig ist auch der Begriff der Normalform: Ein Lambda-Ausdruck N ist in Normalform wenn man keine Beta-Reduktion mehr ausführen kann. Man sagt dass ein Lambda-Ausdruck M Normalform N hat, wenn man M zu N reduzieren kann, und die Normalform ist dann eindeutig. Hier ein Beispiel: Wir beginnen mit dem Lambda-Ausdruck $C_3 Iz$ und stellen fest, dass er als Normalform z hat.

$$\begin{aligned} C_3 Iz = (\lambda f.\lambda x.f(f(fx)))(\lambda x.x)z &\longrightarrow (\lambda x.x)((\lambda x.x)((\lambda x.x)z)) \\ &\longrightarrow (\lambda x.x)((\lambda x.x)z) \\ &\longrightarrow (\lambda x.x)z \longrightarrow z \end{aligned}$$

Haben alle Lambda-Ausdrücke Normalformen? Leider nicht. Betrachte ω auf sich selbst angewandt:

$$\omega\omega = (\lambda x.xx)(\lambda x.xx) \longrightarrow (\lambda x.xx)(\lambda x.xx) \longrightarrow \cdots$$

Egal wie oft wir die Beta-Reduktion anwenden, wir kommen nie bei einem Term an bei dem es nicht mehr weitergeht.

3 Numerische Funktionen

Wie bereits erwähnt zeigte Church dass das Lambda-Kalkül so ausdrucksstark ist wie die Theorie der rekursiven numerischen Funktionen, wobei numerische Funktion hier eine Funktion von den natürlichen in die natürlichen Zahlen meint. Wie kann man ein solche Funktion im Lambda-Kalkül darstellen? Dazu müssen wir zuerst die natürlichen Zahlen definieren. Eine habt ihr schon gesehen, die Drei, und hier ist die allgemeine Definition:

$$C_i = (\lambda f.\lambda x.\underbrace{f(f(\cdots(f}_{i \text{ mal}} x))), \quad i \in \mathbb{N}.$$

Wenn ihr, nur für jetzt, die Variable f als Funktion und die Variable x als einen Wert interpretiert, dann wendet der Lambda-Ausdruck, der die Zahl 42 darstellt, die Funktion f 42 mal auf den Wert x an. All diese Zahlen sind übrigens in Normalform.

Als nächstes können wir definieren was es heißt, dass ein Lambda-Ausdruck F eine numerische Funktion f darstellt: Wenn die Anwendung von F auf die Kodierung einer natürlichen Zahl i sich reduzieren lässt zur entsprechenden natürlichen Zahl des Bildes $f(i)$:

$$\forall i, j \in \mathbb{N}: \quad f(i) = j \iff FC_i \longrightarrow^* C_j$$

Jetzt können wir also Lambda-Ausdrücken benutzen um mit Zahlen zu arbeiten. Für unseren Beweis müssen wir jetzt auch noch Lambda-Ausdrücke nummerieren, also wiederum als Zahlen darstellen. Dazu verwenden wir eine Gödel-Nummerierung G. Wie das genau funktioniert ist gar nicht so wichtig, wir könnten zum Beispiel die Primzahlzerlegung verwenden, so dass die Hochzahl der i-ten Primzahl das Symbol an der i-ten Stelle im Lambda-Ausdruck angibt, wobei verschiedene Hochzahlen verschiedene Symbole darstellen:

$$G[(\lambda x.xx)] = 2^{11} \cdot 3^1 \cdot 5^{17} \cdot 7^{17} \cdot 11^{17} \cdot 13^{13}$$

Diese Zuordnung ist eindeutig und umkehrbar und, wie Hilbert es gesagt hätte, effektiv berechenbar.

Jetzt wo die Lambda-Ausdrücke nur noch Zahlen sind können wir numerischen Funktion darauf loslassen. Zum Beispiel gibt es eine Funktion die zu einer natürlichen Zahl angibt, ob sie einen gültigen Lambda-Ausdruck kodiert. Oder eine Funktion die zwischen natürlichen Zahlen und der Gödel-Nummer des Lambda-Ausdrucks, der diese Zahl darstellt, umwandelt (verwirrend, nicht?). Oder eine Funktion die einen Gödel-kodierten Lambda-Ausdruck Beta-reduziert. Oder eine Funktion die prüft ob ein Gödel-kodierter Lambda-Ausdruck in Normalform ist. Oder eine Funktion die erkennt ob ein Gödel-kodierter Lambda-Ausdruck eine Normalform hat…

$n \mapsto n \in G[\Lambda]?$
$n \mapsto G[C_n]$
$n \mapsto G[N]$, wobei $G^{-1}[n] \longrightarrow N$
$n \mapsto G^{-1}[n] \not\longrightarrow?$
$n \mapsto \exists N \in \Lambda.\ G^{-1}[n] \longrightarrow^* N \not\longrightarrow?$

4 Das unlösbare Problem

Oder vielleicht nicht? Hier sind wir auf den Haken an der ganzen Sache gestoßen: Eine solche Funktion gibt es nicht! Deshalb heißt Churchs Paper auch „An unsolvable problem of number theory“. Lasst mich nun den Beweis skizzieren bevor wir sehen was das mit dem Entscheidungsproblem zu tun hat.

Angenommen, wir hätten ein solche Funktion. Dann hätten wir sicherlich eine Funktion die prüft ob ein gegebener Lambda-Ausdruck als Normalform eine der Zahlen C_i hat, und

welche. Nun gibt es eine Aufzählung A aller Lambda-Ausdrücke mit Normalform – das zeigt Church in seinem Paper. Nun definieren wir die numerische Funktion e wie folgt:

$$e(n) = \begin{cases} i+1, & \text{wenn } A_n C_n \longrightarrow^* C_i \text{ für ein } i \in \mathbb{N} \\ 1, & \text{sonst.} \end{cases}$$

Aufgrund unserer Annahme ist die Funktion e effektiv berechenbar, also gibt es einen Lambda-Ausdruck E der sie darstellt. E hat Normalform, was aus einem hier übersprungenem Lemma folgt. Also ist E irgendwo in der Aufzählung A, etwa $E = A_n$. Der Lambda-Ausdruck EC_n hat als Normalform eine Zahl C_i. Also, per Definition von e, ist $e(n) = i+1$, und dass ist der Widerspruch, den wir brauchen.

$$EC_n \longrightarrow^* C_i \implies A_n C_n \longrightarrow^* C_i \implies e(n) = i+1 \implies EC_n \longrightarrow^* C_{i+1}.$$

5 Das Entscheidungsproblem

Nun wissen wir dass es keine effektive Methode gibt um zu entscheiden ob ein Lambda-Term eine Normalform hat oder nicht. Daraus können wir herleiten dass das Entscheidungsproblem nicht lösbar ist.

Wir nehmen dazu eine beliebige korrekte Logik die stark genug ist um über natürliche Zahlen und rekursive Funktionen und was wir halt so von einer nützlichen Logik erwarten, etwa die in der principia mathematica. In dieser Logik gibt es sicherlich eine Formel die angibt, ob ein Gödel-kodierter Lambda-Ausdruck in Normalform ist, und auch eine Formel die die Beta-Reduktion als Relation darstellt. Die transitive Hülle ist auch darstellbar, also kann man die Aussage, dass ein Lambda-Term eine Normalform hat, als ein Prädikat φ angeben.

$$\models \varphi(m) \iff \exists N \in \Lambda.\, G^{-1}[m] \longrightarrow^* N \not\longrightarrow$$

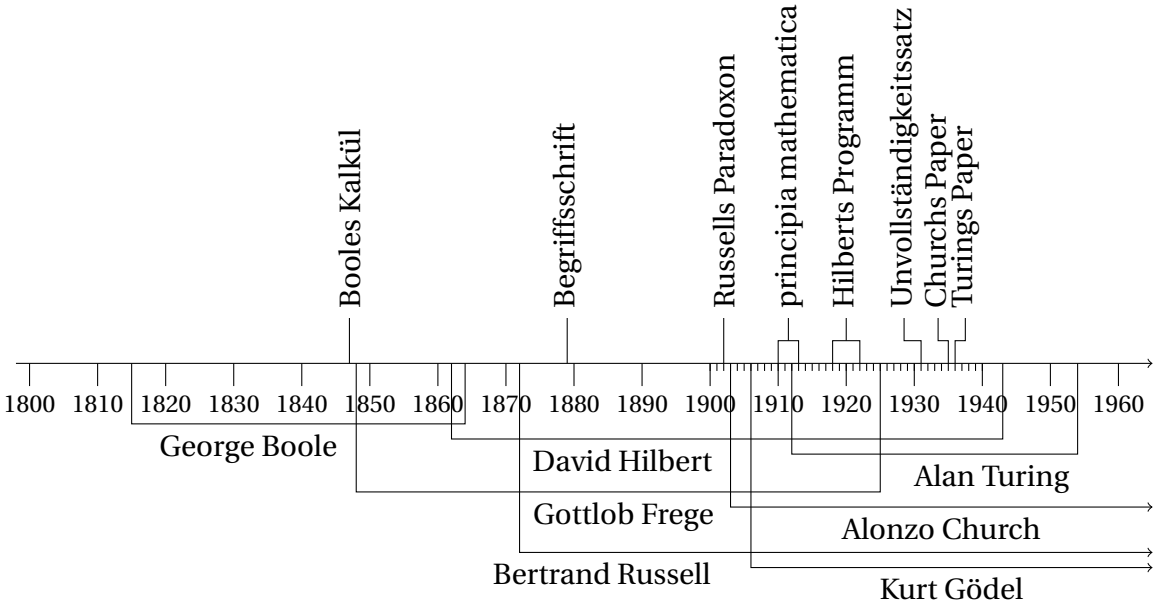
Wenn ein Lambda-Term M Normalform hat, dann ist $\varphi(G[M])$ in der Logik beweisbar, denn ein Beweis kann einfach durch die Zwischenschritte und die Normalform am Ende angegeben werden. Hat M dagegen keine Normalform, dann ist die Aussage $\varphi(G[M])$ nicht beweisbar, denn die Logik soll ja konsistent sein.

Angenommen, das Entscheidungsproblem wäre lösbar. Dann hätten wir jetzt eine effektive Methode um zu entscheiden, of $\varphi(G[M])$ beweisbar ist oder nicht. Demnach hätten wir eine effektive Methode um zu entscheiden ob M eine Normalform hat oder nicht. Und das haben wir ja gerade gesehen, geht nicht.

Daraus folgt dass es keine Hoffnung gibt, Mathematiker durch Algorithmen zu ersetzen und mit diesem schönen Ergebnis ist mein Vortrag am Ende angekommen.

Zeitleiste

Zur Darstellung der Ereignisse habe ich eine Zeitleiste vorbereitet. Die dargestellten Daten sind: Boole 1815-1864, sein Kalkül 1847. Frege 1848-1925, Begriffsschrift 1879. Hilbert 1862-1943, Hilberts Programm 1918-1922. Russell 1872-1970, sein Paradoxon 1902, principia mathematica 1910-1913. Church 1903-1995, Paper 1935. Gödel 1906-1978, Unvollständigkeitssatz 1931, Turing 1912-1954, Paper 1936.



Quellen

- Alonzo Church: “An unsolvable Problem of Elementary Number Theory”, American Journal of Mathematics, Vol. 58, No. 2 (April 1936), 345-363
- Yuri Gurevich: “The Church-Turing Thesis: Story and Recent Progress”, Google Techtalk, June 8, 2009
- Wikipedia natürlich, was denkst du denn?