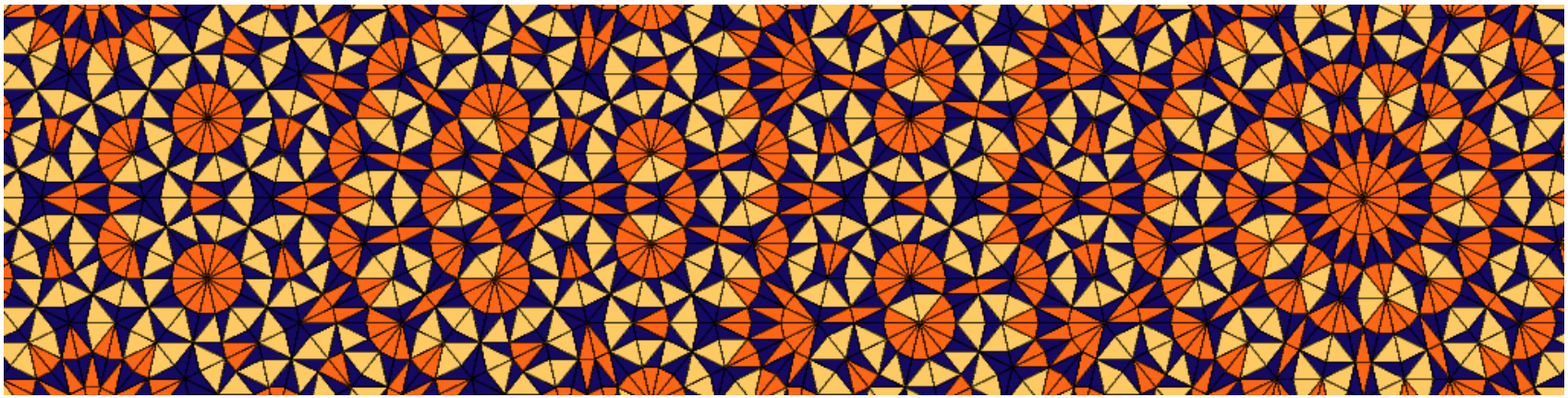
Panorama der Mathematik und Informatik

13: Alan Turing - Turingmaschinen und math. Biologie

Dirk Frettlöh
Technische Fakultät / richtig einsteigen



Recall: 1931 beweist Gödel, dass man nicht alles beweisen kann (Details siehe Vorlesung 12)

Schock. Sind es vielleicht nur ganz exotische Aussagen, die nicht bewiesen werden können? (Vielleicht)

Kann man die Probleme sortieren in "beweisbar" und "nicht beweisbar"? (Nein. Nicht alle.)

Implikation für Informatik: Es gibt (sauber gestellte) Ja-Nein-Fragen, die kein Computer (?! Turingmaschine) entscheiden kann. (Nicht zu verwechseln mit: NP-hart, "nicht effizient" usw.) Beispiele:

- f, g Funktionen (in einer bestimmten Klasse). $f = g$? (Hängt stark von der Klasse ab. Für Polynome entscheidbar.)
- Ein Satz von Polygonen gegeben. Können Kopien davon die Ebene pflastern? (Ohne Lücken, ohne Überlappungen)
- Hat ein Polynom in mehreren Variablen (z.B. $x^3 + 2xy^2 - xy + 7x$) eine ganzzahlige Nullstelle?

Alan Turing

Turing erfand die Turingmaschine, um Gödels Resultat zu verstehen. In der Tat kann man es damit recht fix beweisen:

World's shortest explanation of Gödel's theorem

`http://blog.plover.com/math/Gdl-Smullyan.html`

We have some sort of machine that prints out statements in some sort of language. It needn't be a statement-printing machine exactly; it could be some sort of technique for taking statements and deciding if they are true. But let's think of it as a machine that prints out statements.
In particular, some of the statements that the machine might (or might not) print look like these:

P:x (which means that the machine will print x)
NP:x (which means that the machine will never print x)
PR:x (which means that the machine will print xx)
NPR:x (which means that the machine will never print xx)

For example, NPR:FOO means that the machine will never print FOOFOO. NP:FOOFOO means the same thing. So far, so good.
Now, let's consider the statement NPR:NPR:. This statement asserts that the machine will never print NPR:NPR:.
Either the machine prints NPR:NPR:, or it never prints NPR:NPR:.
If the machine prints NPR:NPR:, it has printed a false statement. But if the machine never prints NPR:NPR:, then NPR:NPR: is a true statement that the machine never prints.
So either the machine sometimes prints false statements, or there are true statements that it never prints.
So any machine that prints only true statements must fail to print some true statements.

Or conversely, any machine that prints every possible true statement must print some false statements too.

Alan Turing (1912-1954)

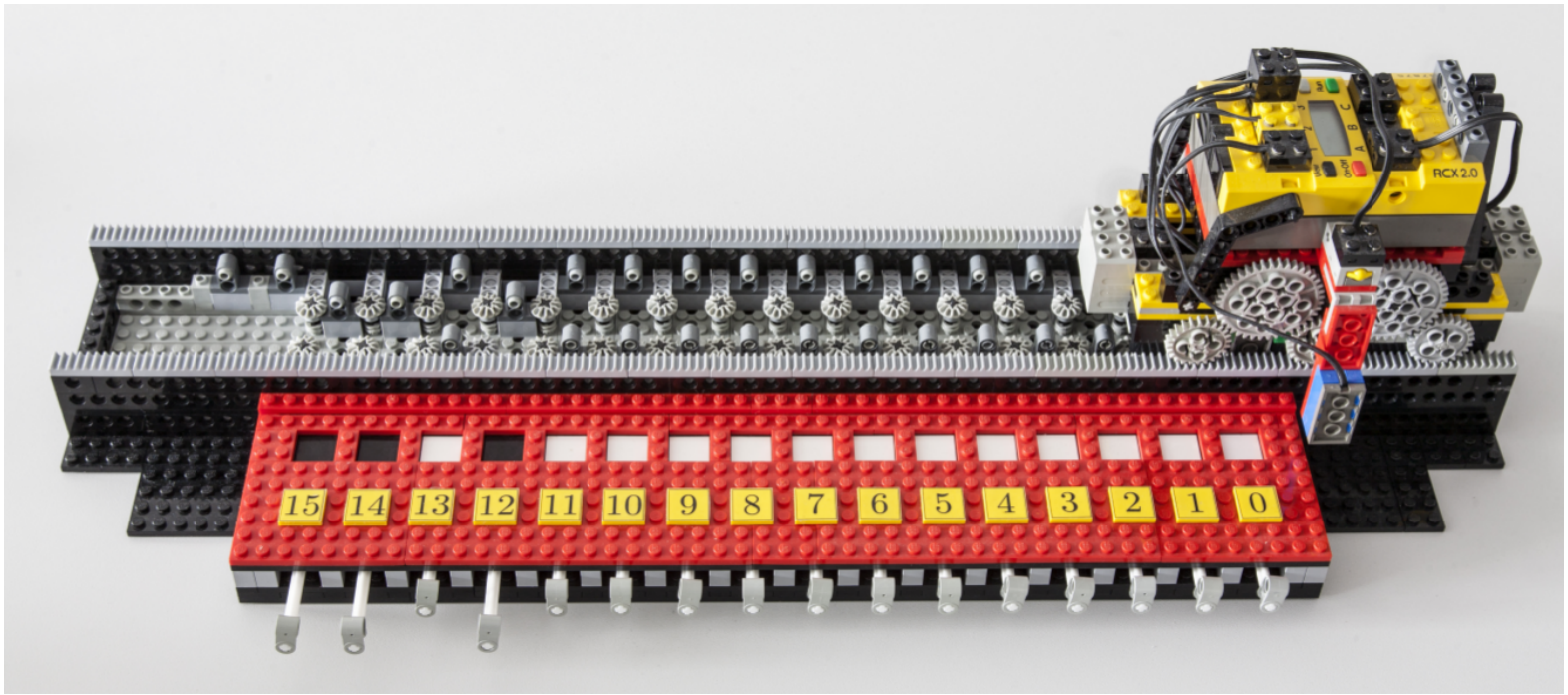
- Schon als Kind sehr talentiert
- Liest (und versteht!) Einsteins Relativitätstheorie
- Beweist mit 22 Jahren eine Version des zentralen Grenzwertsatzes
- Promotion mit 23 Jahren
- Publiziert 1935 (mit 24) "*On Computable Numbers, with an Application to the Entscheidungsproblem*"

Entscheidungsproplem: s.o., "Entscheidbarkeit" im Hilbertprogramm.

Turing formuliert Gödels Beweis einfacher (und erledigt gleichzeitig Hilberts Forderung nach "Entscheidbarkeit").

Dazu musste Turing klären: Was ist ein Algorithmus?

Seine Definition: eine Turingmaschine. Und “berechenbar” oder “entscheidbar” ist alles, was eine Turingmaschine ausrechnen kann.

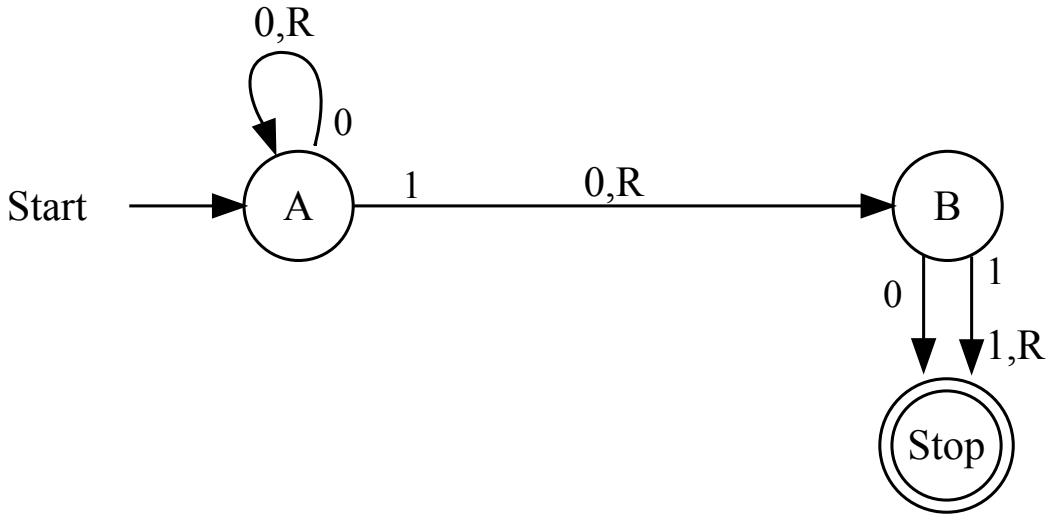


Turingmaschine

Zutaten:

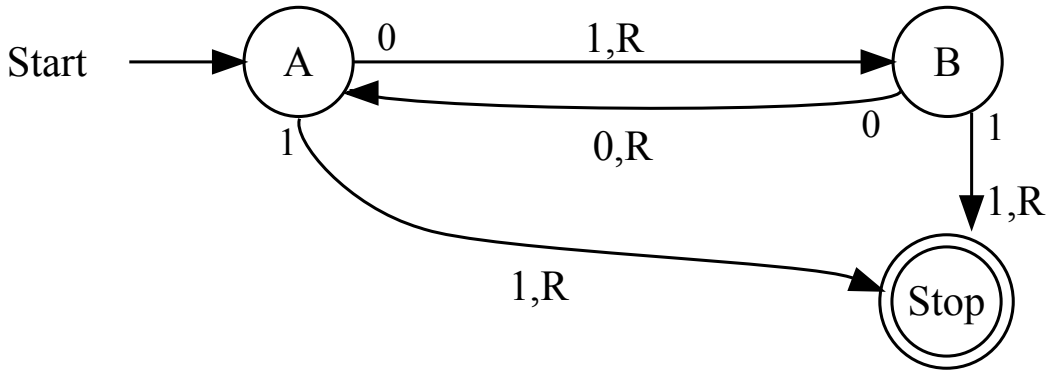
- **Band** aus Zellen $\ldots b_{-1}, b_0, b_1, b_2, \ldots$
- Die Zellen enthalten immer einen von endlich vielen Werten des **Alphabets** $\{0, 1, \ldots\}$.
- Ein **Schreib-/Lesekopf**, der zellenweise über das Band läuft und die aktuelle Zelle lesen und ändern kann
- Ein **Zustandsspeicher**, der einen von endlich vielen Zuständen aus der **Zustandstabelle** $\{A, B, \ldots\}$ annehmen kann. Einer dieser Zustände ist STOP.
- Eine **Aktionstabelle**, in der steht, was als nächstes geschieht, wenn in Zustand x Symbol y vom Band gelesen wird. ("Schreibe z, gehe nach rechts/links, wechsle in Zustand t")

Zu Beginn ist das Band nur mit Nullen beschriftet.



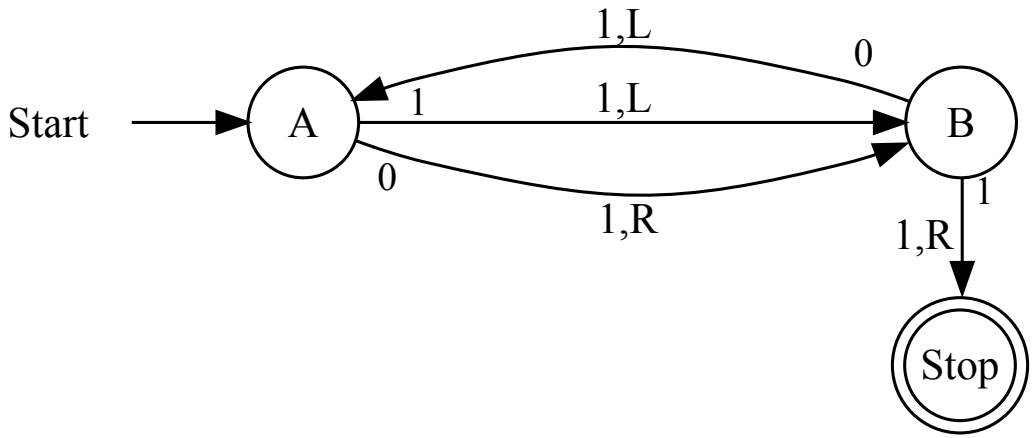
A	0	0	0	0	**0**	0	0	0	0
A	0	0	0	0	0	**0**	0	0	0
A	0	0	0	0	0	0	**0**	0	0
A	0	0	0	0	0	0	0	**0**	0
A	0	0	0	0	0	0	0	0	**0**
A	0	0	0	0	0	0	0	0	0

usw. (hält nie an)



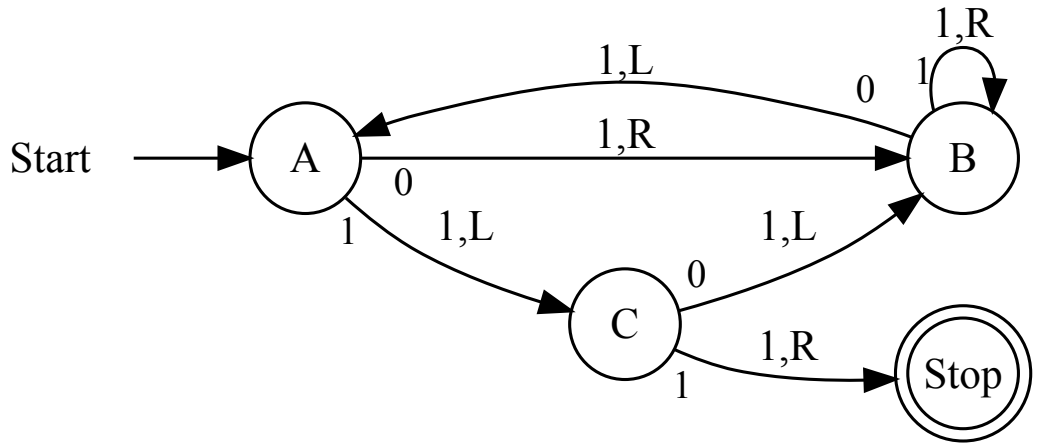
A	0	0	0	0	**0**	0	0	0	0
B	0	0	0	0	1	**0**	0	0	0
A	0	0	0	0	1	0	**0**	0	0
B	0	0	0	0	1	0	1	**0**	0
A	0	0	0	0	1	0	1	0	**0**
B	0	0	0	0	1	0	1	0	1

usw. (hält nie an)



A	0	0	0	0	**0**	0	0	0	0
B	0	0	0	0	1	**0**	0	0	0
A	0	0	0	0	**1**	1	0	0	0
B	0	0	0	**0**	1	1	0	0	0
A	0	0	**0**	1	1	1	0	0	0
B	0	0	1	**1**	1	1	0	0	0

STOP.



A	0	0	0	0	**0**	0	0	0	0	
B	0	0	0	0	1	**0**	0	0	0	
A	0	0	0	0	**1**	1	0	0	0	
C	0	0	0	**0**	1	1	0	0	0	
B	0	0	**0**	1	1	1	0	0	0	
A	0	**0**	1	1	1	1	0	0	0	
B	0	1	**1**	1	1	1	0	0	0	
B	0	1	1	**1**	1	1	0	0	0	
B	0	1	1	1	**1**	1	0	0	0	
B	0	1	1	1	1	**1**	0	0	0	
B	0	1	1	1	1	1	**0**	0	0	
A	0	1	1	1	1	**1**	1	0	0	
C	0	1	1	1	**1**	1	1	0	0	STOP

In seinem Artikel beschreibt Turing ein paar Beispiele, z.B. eine Turingmaschine, die $0101010101010\cdots$ ausgibt (s.o.), oder

$$01011011101111011111011111110111111110111111101111\cdots$$

Eine Turingmaschine kann alles berechnen, was ein moderner Computer (random access machine, von-Neumann-Rechner) berechnen kann.
Umgekehrt heißt jede Maschine, die eine Turingmaschine simulieren kann, **turingvollständig**.

A Turing machine can simulate any type of subroutine found in programming languages, including recursive procedures and any of the known parameter-passing mechanisms
(Hopcroft and Ullman 1979, p157)

"Berechnen": läuft, hält an, Antwort steht auf dem Band.

Daher immer noch nützliches Modell in Theoretischer Informatik.
(Natürlich nicht für Laufzeitanalysen.)

In "*On Computable Numbers, with an Application to the Entscheidungsproblem*" definiert Turing **berechenbare Zahlen** (*computable numbers*).
Das sind die, die mittels Turingmaschinen berechnet werden können.

Moderne **Definition:** (M. Minsky) $x \in \mathbb{R}$ heißt *berechenbar*, falls es zu jedem $n \in \mathbb{N}$ eine Turingmaschine gibt, die die n-te Nachkommastelle von x ausgibt.

Turing listet auf, welche u.a. dazugehören:

Ganze Zahlen, rationale Zahlen, Grenzwerte von "berechenbar konvergenten" Folgen, Grenzwerte von unendlichen Reihen mit berechenbaren Summanden, z.B.

$$e = 1 + 1 + \frac{1}{2} + \frac{1}{6} + \frac{1}{24} + \cdots, \quad \pi = 4(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \cdots)$$

...grob: alle Zahlen, die man sich vorstellen kann. Der Witz ist:

Satz: Die berechenbaren Zahlen $\mathbb{B}$ sind abzählbar: $|\mathbb{B}| = |\mathbb{N}|$.

Denn: Offenbar ist es einfach, jede Turingmaschine als natürliche Zahl zu kodieren (Alphabet, Zustände, Aktionstabelle).

Jede Turingmaschine entspricht einem $n \in \mathbb{N}$, also gibt's nur $\mathbb{N}$ viele Turingmaschinen.

Zu jeder berechenbaren Zahl gibt's mindestens eine Turingmaschine.

Die meisten Zahlen in $\mathbb{R}$ wird man niemals sehen!

Immer noch im selben Artikel zeigt Turing auch die Unlösbarkeit des Entscheidungsproblems.

Turing erkennt dass es eine “universelle Turingmaschine” gibt. Die spuckt nacheinander alle beweisbaren Formeln aus.

Wird die Maschine jemals NPR:NPR: schreiben?

Nein, s.o.: Falls ja, wäre es falsch. Falls nein, ist’s nicht beweisbar.

Alan Turing

- 1936-1938 in Princeton (Uni, nicht IAS)
- ab 1939 Arbeit an der Entschlüsselung deutscher Codes in Bletchley Park
- dazu: (Weiter-)Entwicklung mechanischer Rechengeräte ("bombs", polnisch "Bomba", Biuro Szyfrów)
- Prinzipielle Entschlüsselung der *Enigma*, damit des deutschen U-Boot-Funks
- Ressourcenmangel, dann 18.11.1941: "ACTION THIS DAY. Make sure they have all they want on extreme priority and report to me that this has been done." (W. Churchill)
- Effektive Entschlüsselung der *Enigma*, damit des deutschen U-Boot-Funks (dazu später mehr: Kryptographie)

Aus der Arbeit in Bletchley Park heraus entstand auch der erste (programmierbare, turingvollständige, elektrische) Computer: Colossus (1945)

Naja, Konrad Zuses Z3 (1941): programmierbarer, turingvollständiger, elektrischer Computer. Ohne IF-Anweisung.

(Siehe wikipedia, Nixdorf-Museum Paderborn)

Ab dann gab's ein Wettrennen um mehr, bessere, schnellere ... Computer, in Unis, Firmen, Militär ... das die USA gewannen.

Zu Alan Turing (1912-1954):

Arbeitet nach dem 2. Weltkrieg zwei Jahre weiter an Computern.

Ging zurück an eine Uni.

- Turingtest
- LU-Zerlegung von Matrizen
- Mathematische Biologie ("wie kommt der Leopard zu seinen Flecken?")

1952 Verurteilung, 1954 Tod.