"Hilberts Albtraum" (The answer may *not* be out there!)

Jonas Betzendahl @jbetzend

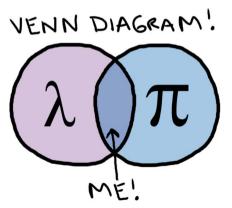


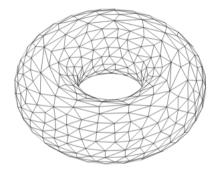




#### Mein Thema

Wie kann ich einem Computer beibringen, mit mathematischen Beweisen umzugehen?





Mathematik ist eine Wissenschaft fast wie jede andere...

<sup>&</sup>quot;3D Torus ..." by GDJ, www.openclipart.org, licensed CC-0

Math is the	reverse of	comedv.	The .	anti-ioke.		

the right punchline.

We'll tell you the punchline first, then laboriously explain to you why it was

Joseph Maher, College of Staten Island

GIVEN THE PACE OF TECHNOLOGY, I PROPOSE WE LEAVE MATH TO THE MACHINES AND GO PLAY OUTSIDE.

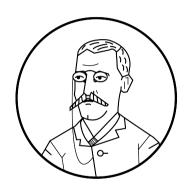


#### David Hilbert

Mathematiker (1862-1943) aus Königsberg, bekannt für seine Grundlagenforschung und Problemsammlungen.

#### Hilberts Traum:

Ein Computer, dem ich eine beliebige mathematische Aussage reichen kann und der mir sagt, ob sie stimmt.



<sup>&</sup>quot;David Hilbert" by Matthew Leadbeater, licensed CC-BY-NC, 2016

Das ist leicht zu schaffen, wenn wir das *Halteproblem* lösen!

(z.B. durch Auflistung aller Folgerungen aus den Axiomen)

#### Halteproblem:

Gegeben nur den Quellcode und den Input eines Programms, sage vorher ob dieses Programm jemals "fertig wird" oder endlos weiter läuft.

#### extract num-

ber, and, incr (destination, source) int "destination: unsigned char "hourse: I entract name her identination, "sourcel: "source += 2:1 tithdef EXTRACT, MAC-ROS Runder EXTRACT NUMBER, AND INCR Adefine EXTRACT, NUM BER AND INCROSET ord Leutract number and incrobbert Burd Bendit? not EXTRACT. MACROS 1/ Aend? /\* DEBUG 1/30,\*\* HDEBUG is defined. Repex prints many voluminous messages about what it is doing lift the variable "debug" is nagress). If linked with the main program in 'inequal', you can enter patterns and strings interactively. And if linked with the main program in 'main,c' and the other test files, you can run the alreach continue tests. N. #8646 PMBUS P. We use standard LID for debusining. N. Bindada catific by All is useful to test things that "exect" be true when debugging. Witherland consent its static test debug = 0: #define DEBLIG: STATEMENT(e) o #define DEBLIG: PRINT (b) if (debug) printf (c) #define DERLIG PRINTS(x), x2) of Orleans) month (x), x2) educing DERLIG PRINTS(x), x2, x3) of Inhabitation prints (x1, x2, x3) #define DEBUG\_PRINT4(x1, x2, x8, x4) if (debug) printf (x1, x2, x3, x4) #define DE-BUG PRINT COMPLED PATTERNIp. s. e/l if idebug) print, partial compiled gattern (s, e) #define DE-BUG PRINT DOUBLE STRINGIN (1, cy1, s2, cy2) \ \( \text{if (debug) print, double, string (w, c1, cy1, s2, cy2)} \) entern unid prints har? If Print the featmen in human couldable from 17 unid print featmen (featmen) char \*fastmap: ( unsigned was, a range = 0; unsigned i = 0; while (i < 0! << EFTEWIDTHD ( if !fastmap(i++)) I was a nance = 0 printchar () - 11: while () < () << EFTEWDD19 66 fastmap(i) () was a nance = 1: i++: ) if lwas, a. range) ( printf (\*\*); print(har (i - 1); ) )) putchar (\*\*); ) /\* Print a compiled pattern string in human-readable form, starting at the START pointer into it and ending just before the pointer END. "I word print, partial, compalled, pattern (start, end) unsigned char "start, unsigned char "end) (interest record) unsigned that 'to a start unsigned that 'nord a end if start as NIE 131 print ("indiffer"); return 1," Loop over pattern commands. \*/ while (p < pend) ( switch (ise, opcode, () \*p++) ( case no, op; printf ("/no, op"); break case exacts: recnt = "a++; printf ("/exacts/hef", more); do ( putcher f/h; printcher ("p++); ) while (-mont): break case start, memory mont = "p++; printf ["/start\_memory/tid/fiel", mont. The all books case aton memory more - the a might Citizen memory biddlied most the all break case duplicate: printf ("/duplicate/%d", "p.i.-i.); break; case awehar; printf ("/arachar"); break case charset case charset, not i register intic printf ("/charsetlis") (re-opcode til flo-Till an charact not 1" not": "It assert to + "n < pends for lc = 0; c < "n; c++1 luminoed bit: participated charging that a politic of participate (20) for this probability that a BYTEWOOD A better if F Imap, byte & CLoc bits printshar is "BYTERROTH + bits I p += 1 + "p; break I case beqline printf ("thenline"), brook case ending mintf ("landline") brook case on failure kerner entract, number, and increifment field printf from foliage kernel(Midf. recent). break case on failure keen string keen estract number and incretement Api point O'ce failure keep string jump/0/hig! mont: break case dummy failure jump extract mumber and incr@mont.&pt printf ("idummy falluse tump/\$7%d", mont); break case path, dummy, failure: printf ("(path, dummy, failure"), break; case mayhe non turns extract number and incoffigures feel result C/maybe non jump/0/%df month break case non fallure

Manchmal ist es sehr leicht zu sehen, ob ein Programm jemals halten wird oder nicht:

```
-- Dieses Programm hält quasi sofort

main :: IO ()

main = print $ plus (3,4)

where

plus :: (Int, Int) -> Int

plus (x,y) = x + y
```

Manchmal ist es sehr leicht zu sehen, ob ein Programm jemals halten wird oder nicht:

```
-- Dieses Programm hält quasi sofort

main :: IO ()

main = print $ plus (3,4)

where

plus :: (Int, Int) -> Int

plus (x,y) = x + y
```

```
-- Dieses Programm läuft "für immer"

main :: IO ()

main = forever $ print "lol, infinite loop!"
```

Manchmal ist es aber auch nahezu unmöglich!

```
-- Nobody knows if this ever halts...
main :: IO ()
main = do let results = filter isPerfect [1,3...]
          case results of
            [] -> print " No odd perfect numbers!"
            -> print "Yes odd perfect numbers!"
divisors :: Int -> [Int]
divisors n = filter (\x -> n \rem \x == 0) [1..n]
isPerfect :: Int -> Bool
isPerfect n = (sum . divisors) n == n + n
```

#### Alan Turing



Mathematiker (1912-1954) aus London. Half in Bletchley Park, den dt. *Enigma*-Code zu lösen.

Bewies in seiner Doktorarbeit, dass das Halteproblem *nicht lösbar sein kann* (zumindest im allgemeinen Fall)!

#### Alan Turing



Mathematiker (1912-1954) aus London. Half in Bletchley Park, den dt. *Enigma*-Code zu lösen.

Bewies in seiner Doktorarbeit, dass das Halteproblem *nicht lösbar sein kann* (zumindest im allgemeinen Fall)!

Aber nehmen wir mal an, es wäre lösbar...

# Ein magisches Halte-Orakel (1)



#### Ein magisches Halte-Orakel (2)



# Ein magisches Halte-Orakel (3)



#### Ein magisches Halte-Orakel (4)

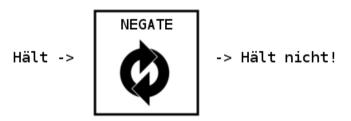


# Negierer (1)



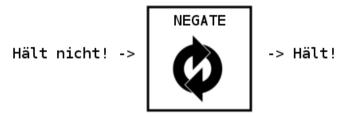
Dazu kommt ein Negierer, der ein Haltergebnis umdreht!

# Negierer (2)



Dazu kommt ein Negierer, der ein Haltergebnis umdreht!

# Negierer (3)



Dazu kommt ein Negierer, der ein Haltergebnis umdreht!

# Duplizierer (1)



Wir haben auch einen Duplizierer, der seine Eingabe verdoppelt!

# Duplizierer (2)



Wir haben auch einen Duplizierer, der seine Eingabe verdoppelt!

# Duplizierer (3)

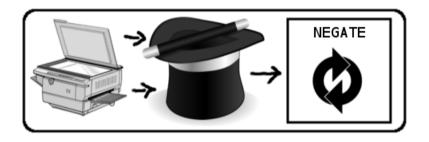
Wir haben auch einen Duplizierer, der seine Eingabe verdoppelt!

#### Unite and conquer! (1)



Manche ahnen es vielleicht schon! Alles das können wir jetzt zusammen stecken und hintereinander schalten!

# Unite and conquer! (2)



Insbesondere können wir diese Aneinanderreihung als eine Maschine betrachen. Nennen wir sie X.

# Unite and conquer! (3)



Insbesondere können wir diese Aneinanderreihung als eine Maschine betrachen. Nennen wir sie X.

# Going Meta! (1)



Jetzt nur keine Panik bekommen! Aber was passiert, wenn wir den Quellcode von X an X geben?

# Going Meta! (2)



Der interessante Teil der Frage ist, was das Orakel ausspuckt. Hier gibt es aber überhaupt nur zwei Möglichkeiten!

Fall 1: Orakel sagt "Hält!"



Entweder das Orakel sagt das X mit X als Inpt hält.

Das geht aber nicht, weil dann der Negierer dafür sorgt, dass es *nicht* hält.

Fall 2: Orakel sagt "Hält nicht!"



Oder das Orakel sagt X mit X als Inpt hält *nicht*!

Das bringt aber das gleiche Problem: dank des Negierers hält X eben doch!

#### Widerspruch



Was bleibt uns von unserer erdachten magischen Maschine? Jeder mögliche Weg führt zum Widerspruch!

#### Widerspruch



Was bleibt uns von unserer erdachten magischen Maschine? Jeder mögliche Weg führt zum Widerspruch!

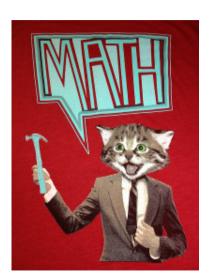
Damit ist bewiesen: so ein Orakel *kann* es nicht geben! Das Halteproblem ist (im Allgemeinen) unlösbar!

#### Widerspruch

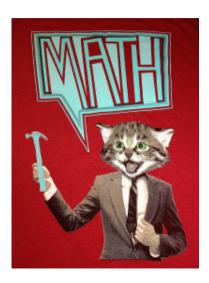


Was bleibt uns von unserer erdachten magischen Maschine? Jeder mögliche Weg führt zum Widerspruch!

Damit ist bewiesen: so ein Orakel *kann* es nicht geben! Das Halteproblem ist (im Allgemeinen) unlösbar!

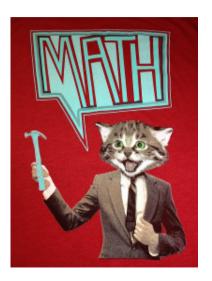


Was haben wir heute gelernt?



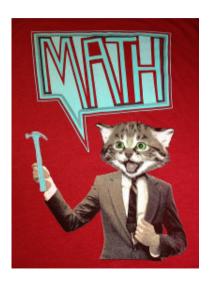
Was haben wir heute gelernt?

• Die Welt ist groß und ungewiss!



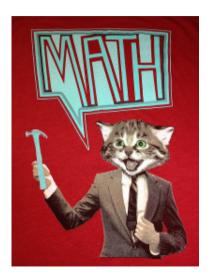
#### Was haben wir heute gelernt?

- Die Welt ist groß und ungewiss!
- Mathe macht Spaß!



#### Was haben wir heute gelernt?

- Die Welt ist groß und ungewiss!
- Mathe macht Spaß!
- Wir können jetzt das Halteproblem erklären und wissen, dass es keine allgemeine Lösung gibt.



#### Was haben wir heute gelernt?

- Die Welt ist groß und ungewiss!
- Mathe macht Spaß!
- Wir können jetzt das Halteproblem erklären und wissen, dass es keine allgemeine Lösung gibt.

Bitte bleibt neugierig!