Proof of Theorem 4.5.1 [LP pg 224.]

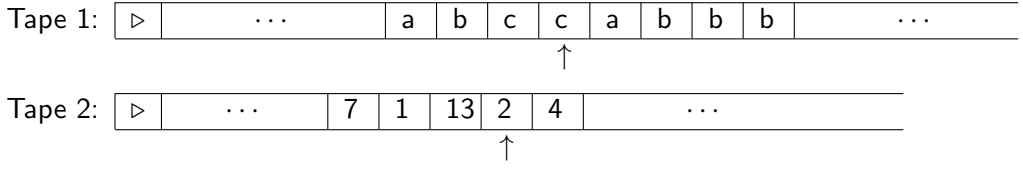
Main idea:

Simulate all computations until you get a halt (if ever).

At a given configuration, say $(q, u\underline{a}v)$. The next step considers only state q and symbol a. The maximum number of next steps is

$$r = |K| \cdot (|\Sigma| + 2)$$

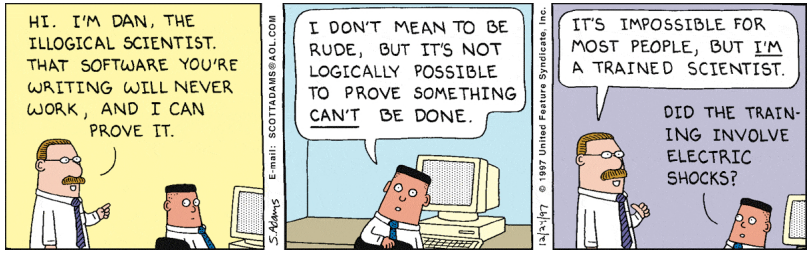
Let M_d be the "deterministic version" of M.



Proof of Theorem 4.5.1, continued.

Define M' with three tapes: One for the original input, one for the tape of the current simulation of M, and one for the current hint tape for M_d. Algorithm for M':

copy input onto the simulation tape
put 1 onto the hint tape
L: Operate like M_d
if you ever halt, then great!
if you run out of hints,
copy original input back to the simulation tape
put the *lexicographically next* hint on the hint tape
goto L



https://dilbert.com/search_results?terms=Illogical+Scientist

Undecidability has been so much a part of the culture of computer science since its beginnings, that it is easy to forget what a curious fact it is. Strictly speaking, once we accept the identification of problems with languages and algorithms as Turing machines, there are trivial reasons why there must be undecidable problems: There are more languages (uncountably many) than ways of deciding them (Turing machines). Still, that such problems exist so close to our computational ambitions was a complete surprise when it was first observed in the 1930's. Undecidability is in some sense the most lethal form of complexity.

C Papadimitriou, Computational Complexity, *pg 59*

§5.2 **The Universal Turing Machine**

Design universal Turing machine U and specify a language of descriptions of Turing machines such that $U(\text{“}M\text{”}\ \ \text{“}w\text{”}) = \text{“}M(w)\text{”}$

- Use strings to represent tape symbols. a means, “this is a tape symbol.”
- Use strings to represent states. q means, “this is a state.”
- Have three tapes:
  1. To simulate M's tape
  2. To store the description of M
  3. To simulate M's state
- Scan the description of the transition function until an appropriate transition is found.

§5.3 **The Halting Problem**

$H \in RE$, but $H \notin R$; hence although $R \subseteq RE$, $RE \neq R$.

- Suppose we have a program `halts`
- Build the program `diagonal`:

  `diagonal`(X) :
  a: if `halts`(X, X) then goto a else halt

- Does `diagonal(diagonal)` halt or diverge?

  `diagonal(diagonal)` halts $\longrightarrow$ `halts(diagonal, diagonal)` $\longrightarrow$ `diagonal(diagonal)` diverges

  `diagonal(diagonal)` diverges $\longrightarrow \sim$ `halts(diagonal, diagonal)` $\longrightarrow$ `diagonal(diagonal)` halts

  This kind of argument should be familiar not only from your past exposure to computer science, but also from general twentieth century culture. *LP, pg 251*

Thm 7.23. $(0,1)$ is uncountable.

Proof. *Suppose* $(0,1)$ *is countable. Then there exists a one-to-one correspondence* $f : \mathbb{N} \to (0,1)$. *We will use* f *to give names to the all the digits of all the numbers in* $(0,1)$, *considering each number in its decimal expansion, where each* $a_{i,j}$ *stands for a digit.:*

$$\begin{aligned} f(1) &= 0.a_{1,1}a_{1,2}a_{1,3}\ldots a_{1,j}\ldots \\ f(2) &= 0.a_{2,1}a_{2,2}a_{2,3}\ldots a_{2,j}\ldots \\ &\vdots \\ f(x) &= 0.a_{x,1}a_{x,2}a_{x,3}\ldots a_{x,j}\ldots \\ &\vdots \end{aligned}$$

Now construct a number $d = 0.d_1d_2d_3\ldots d_i\ldots$ *as follows*

$$d_i = \begin{cases} 1 & \text{if } a_{i,i} \neq 1 \\ 2 & \text{if } a_{i,i} = 1 \end{cases}$$

Since $d \in (0,1)$ *and* f *is onto, there exists an* $x \in \mathbb{N}$ *such that* $f(x) = d$. *Moreover,*

$$f(x) = 0.a_{x,1}a_{x,2}a_{x,3}\ldots a_{x,x}\ldots$$

so

$$d = 0.a_{x,1}a_{x,2}a_{x,3}\ldots a_{x,x}\ldots$$

by substitution. In other words, $d_i = a_{x,i}$, *and specifically* $d_x = a_{x,x}$. *However, by the way that we have defined* d, *we know that* $d_x \neq a_{x,x}$, *a contradiction. Therefore* $(0,1)$ *is not countable.* $\square$

Gödel's Theorem appears as Proposition VI in his 1931 paper "On Formally Undecidable Propositions in Principia Mathematica *and Related Systems I." It states [paraphrased]:*

All consistent axiomatic formulations of number theory include undecidable propositions.

Gödel had the insight that a statement of number theory could be about a statement of number theory, if only numbers could somehow stand for statements. The grand conclusion: That the system of [Russel and Whitehead's] Principia Mathematica *is "incomplete"—there are true statements of number theory which its methods of proof are too weak to demonstrate.*

D Hofstadter, Gödel, Escher, Bach, *pg 17–18, abridged*

But if Principia Mathematica *was the first victim of this stroke, it was certainly not the last. The phrase "and Related Systems" in the title of Gödel's article is a telling one; for if Gödel's result had merely pointed out a defect in the work of Russell and Whitehead, then others could have been inspired to improve upon* P.M. *and to outwit Gödel's Theorem. But this was not possible: Gödel's proof pertained to* any *axiomatic system which purported to achieve the aims which Whitehead and Russell had set for themselves. In short, Gödel showed that provability is a weaker notion than truth, no matter what axiomatic system is involved.*

Therefore Gödel's Theorem had an electrifying affect upon logicians, mathematicians, and philosophers interested in the foundations of mathematics, for it showed that no fixed system, no matter how complicated, could represent the complexity of the whole numbers. Modern readers may not be as nonplussed by this as readers of 1931 were, since in the interim our culture has absorbed Gödel's Theorem, along with the conceptual revolutions of relativity and quantum mechanics, and their philosophically disorienting messages have reached the public, even if cushioned by several layers of translation (and obfuscation). There is a general mood of expectation, these days, of "limitive" results—but back in 1931, this came as a bolt from the blue.

D Hofstadter, Gödel, Escher, Bach, *pg 19, abridged*

H is *complete* for the class RE.

If $H \in R$, then $R = RE$. (But it's not.)

Any other RE language can be reduced to H.

Theorem 5.3.1: The language H is not recursive; therefore, the class of recursive languages is a strict subset of the class of recursively enumerable languages.

Theorem 5.3.2: The class of recursively enumerable languages is not closed under complement.

§5.4. **More undecidable problems**

We have a beachhead in $RE - R$. What else can we find?

To show that L is undecidable:

- Suppose L is decidable
- Use the machine that decides L to build a machine that decides the halting problem
- Conclude (by contradiction) that L is undecidable

If $L \in R$, then $H \in R$.
$H \notin R$
Therefore $L \notin R$