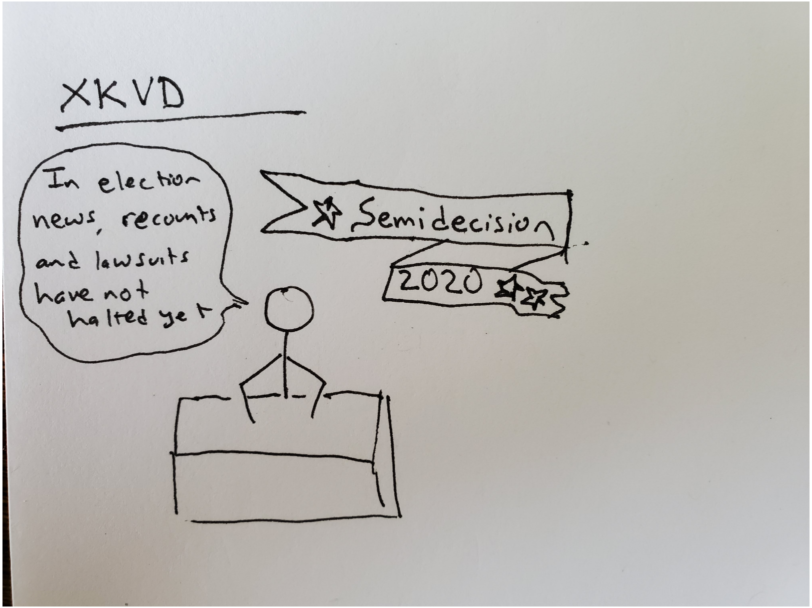
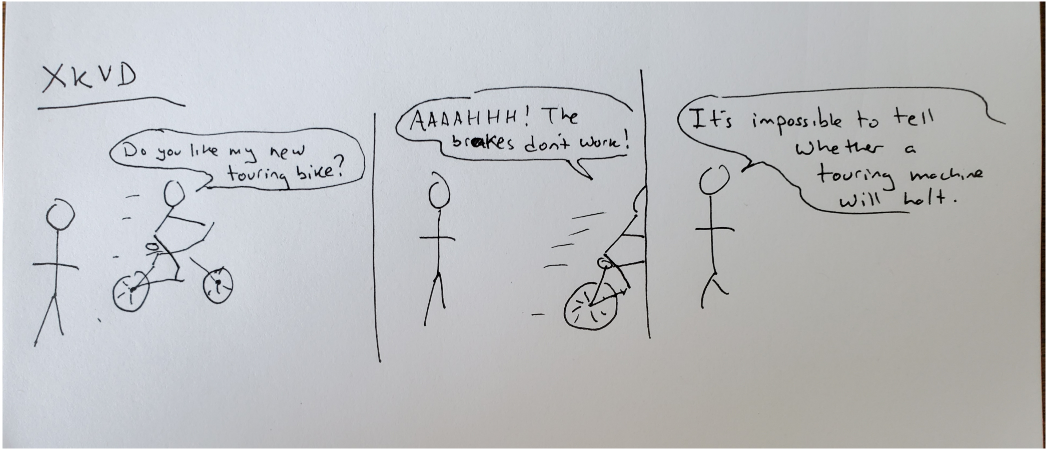

XKVD
In election news, recounts and lawsuits have not halted yet
Semidecision
2020


XKVD
Do you like my new touring bike?
AAAAHHH! The brakes don't work!
It's impossible to tell whether a touring machine will halt.

Schedule (recent and immanent)

Date	Reading	In class
Wed, Nov 13	4.2. Computing with TMs 4.3 Extensions to TMs 4.4 RAM TMs	Through definition of semidecide
Fri, Nov 15	4.5 Nondet. TMs	Through 4.5
Mon, Nov 18	5.1 C-T Thesis 5.2 Universal TMs 5.3 The halting problem 5.4 Undecidable problems	Through 5.3
Wed, Nov 20	5.6 Tiling 5.7 Properties of rec langs	~~5.(4,6,7)~~ 5.4
Fri, Nov 22	6 (whole chapter)	5.(4,6,7), definitions from 6.(1 & 2)

§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 $L \notin R$

More detailed format for proving that some language L is undecidable:

- Choose a known undecidable problem/language L_1.
- Suppose a machine M decides L.
- Define τ, a function from L_1 to L.
- Show that τ is recursive (decidable/computable).
- Show that $w \in L_1$ iff $\tau(w) \in L$. That is, show that the machine made by composing τ and M decides problem L_1, which is absurd.
- Conclude (by contradiction) that M does not exist, that is, that L is undecidable.

5.4.2.a. NO.

Short answer: Suppose such a Turing machine existed. Then suppose we have a machine M and input w. Make a machine that modifies the input M so that all halt states in M transition to a new state q. Then use the machine suggested here to determine if this modified M reaches state q. This would solve the halting problem.

Long answer:

> **Proof.** *We will prove that this problem is undecidable by reducing the halting problem to it.*
> *Suppose there exists a machine M_1 that decides the language of Turing machine, state, string triples (M, q, w) such that M reaches state q when given input w.*

Long answer/proof for **5.4.2.a**, *continued*

Let M_2 be the Turing machine that operates as follows: When given the description of a machine M and input w, M_2 constructs the description of a machine M' such that M' is like M except that it has one more state q, and all the transitions in M that would move to a halting state are changed so that they now transition to q. Then M_2 acts like M_1 on the description of M', q, and w.

Note that by how we defined M_2, it must be that M_2 accepts M, w if and only if M_1 accepts M', q, w.

Further, M_2 decides the halting problem: Suppose a machine M halts on input w. Then the machine M' that M_2 constructs will reach state q on input w, and so M_1 and therefore M_2 will accept it. Next suppose M does not halt on input w. Then the machine M' will never reach state q, and so M_1 and therefore M_2 will reject it.

Since it is impossible for a machine to decide the halting problem, M_2 cannot exist, and therefore M_1 cannot exist. Thus this problem is undecidable. $\square$

5.4.2.b. NO.

Short answer: If we had such a machine we could use it to decide the problem in part a by setting p to the start state.

Long answer:

Proof. *We will prove that this problem is undecidable by reducing the problem in part a to it.*
Suppose there exists a machine M_1 *that decides the language of Turing machine, state, state* (M, p, q) *triples such that there is a configuration with with state* p *that yields a configuration with state* q*.*

Long answer/proof for **5.4.2.b**, *continued*

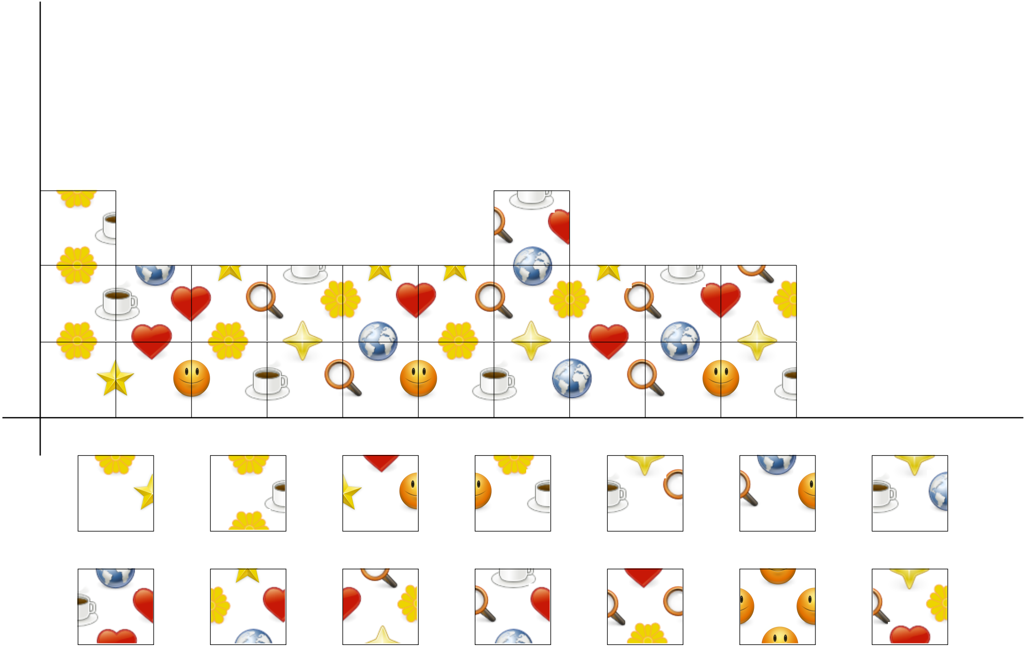
Let M_2 be the Turing machine that operates as follows: When given the description of a machine M, a state q, and a string w, M_2 constructs the description of a machine M' such that M' is like M except that it has a new start state s. (Let s_0 be the start state of M.) When M' is in state s, it erases whatever is on its tape and writes w in its place. Then it moves its head to the beginning and transitions to state s_0; from then on, M' operates like M.

After constructing M', M_2 also adds the description of s and q on the tape and then acts like M_1 does on its input; in other words, it gives (M', s, q) as input to M_1.

Note that by how we defined M_2, it must be that M_2 accepts (M, q) if and only if M_1 accepts (M', s, q).

Further, M_2 solves the problem described in part a: Suppose a machine M reaches state q starting with string w. Then the machine M' that M_2 constructs will reach q from state s. Next suppose a machine M never reaches state q starting with string w. Then the machine M' that M_2 constructs will never reach q from state s.

Since it is impossible for a machine to decide the problem in part a, M_2 cannot exist, and therefore M_1 cannot exist. Thus this problem is undecidable. $\square$



Definition 6.1.1: A Turing machine M is **polynomially bounded** if

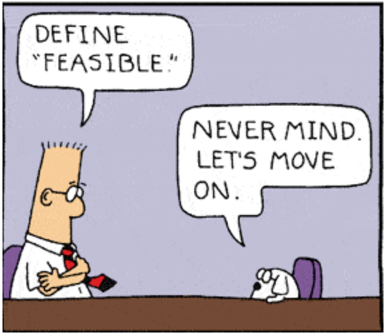
$\exists\ p(n)$, a polynomial function such that
$\quad\forall\ x \in \Sigma*$
$\quad\quad\forall\ C \in$ (set of configurations), either
$\quad\quad\quad C$ is unreachable from $(s, \triangleright\underline{\sqcup}w)$, or
$\quad\quad\quad (s, \triangleright\underline{\sqcup}w) \vdash_M^k C$, where $k \leq p(|x|)$

A language is **polynomially decidable** if

$\exists\ M$, a Turing machine that decides the language, such that
$\quad\exists\ p(n)$, a polynomial function such that
$\quad\quad\forall\ x \in \Sigma*$
$\quad\quad\quad\forall\ C \in$ (set of configurations), either
$\quad\quad\quad\quad C$ is unreachable from $(s, \triangleright\underline{\sqcup}w)$, or
$\quad\quad\quad\quad (s, \triangleright\underline{\sqcup}w) \vdash_M^k C$, where $k \leq p(|x|)$

LP pg 276

§6.2. The class of polynomially decidable languages is denoted $\mathcal{P}$. Why is polynomial time used as a measure of tractability/feasibility?



Scott Adams, 1994