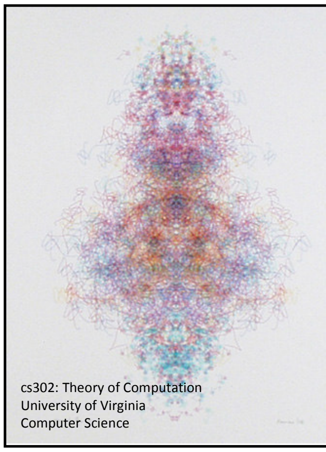


cs302: Theory of Computation
University of Virginia
Computer Science

Lecture 16: Universality and Undecidability

PS4 is due now

Some people have still not picked up Exam 1! After next week Wednesday, I will start charging "storage fees" for them.

David Evans
http://www.cs.virginia.edu/evans

Menu

- Simulating Turing Machines
- Universal Turing Machines
- Undecidability

Proof-by-Simulation = Proof-by-Construction

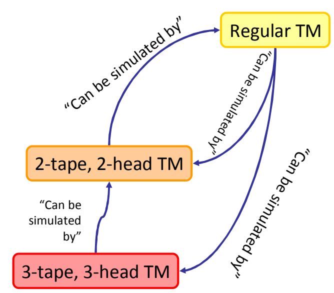
To show an **A** (some class of machines) is as powerful as a **B** (some class of machines) we need to show that for any **B**, there is some equivalent **A**.

Proof-by-construction:
Given any $b \in$ **B**, construct an $a \in$ **A** that recognizes the same language as b.

Proof-by-simulation:
Show that there is some **A** that can simulate any **B**.

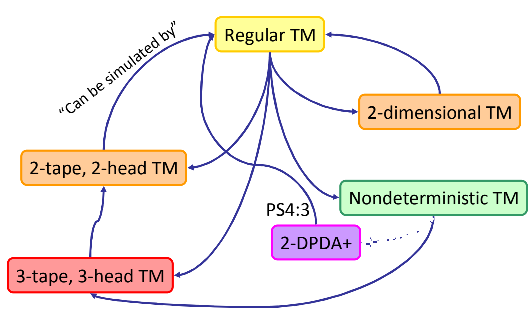
Either of these shows:
languages that can be recognized by a **B** $\subseteq$ languages that can be recognized by an **A**

TM Simulations



If there is a path from *M* to Regular TM and a path from Regular TM to *M* then *M* is equivalent to a Regular TM

TM Simulations



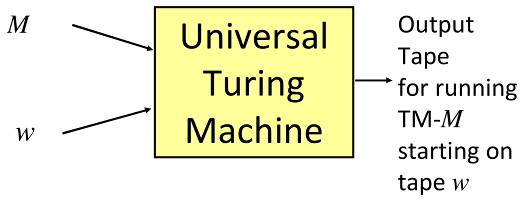
Can a TM simulate a TM?

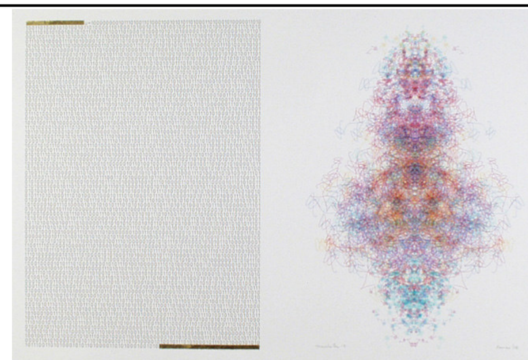
Yes, obviously.

Can *one* TM simulate *every* TM?

An Any TM Simulator

Input: < Description of some TM *M*, *w* >
Output: result of running *M* on *w*



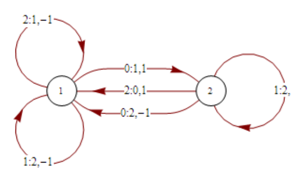


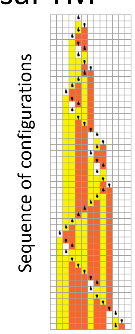
Manchester Illuminated Universal Turing Machine, #9
from http://www.verostko.com/manchester/manchester.html

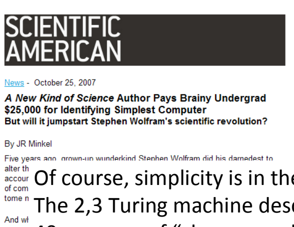
Universal Turing Machines

- People have designed Universal Turing Machines with
  - 4 symbols, 7 states (Marvin Minsky)
  - 4 symbols, 5 states
  - 2 symbols, 22 states
  - 18 symbols, 2 states
  - 2 states, 5 symbols (Stephen Wolfram)
- November 2007: 2 state, 3 symbols

2-state, 3-symbol Universal TM






SCIENTIFIC AMERICAN

News - October 25, 2007

***A New Kind of Science* Author Pays Brainy Undergrad $25,000 for Identifying Simplest Computer**
But will it jumpstart Stephen Wolfram's scientific revolution?

By JR Minkel

Five years ago, grown-up wunderkind Stephen Wolfram did his darnedest to alter th… accour… of com… tome n…

And wh…

Alex Smith, University of Birmingham

Of course, simplicity is in the eye of the beholder. The 2,3 Turing machine described in the dense new 40-page proof "chews up a lot of tape" to perform even a simple job, Smith says. Programming it to calculate 2 + 2, he notes, would take up more memory than any known computer contains. And image processing? "It probably wouldn't finish before the end of the universe," he says.

Rough Sketch of Proof

None of these steps involve universal computation themselves:

System 0 (the claimed UTM)
can simulate System 1
which can simulate System 2
which can simulate System 3
which can simulate System 4
which can simulate System 5
which can simulate any 2-color cyclic tag system
which can simulate any TM.

See http://www.wolframscience.com/prizes/tm23/TM23Proof.pdf for the 40-page version with all the details…

Enumerating Turing Machines

- Now that we've decided how to describe Turing Machines, we can number them
- TM-5023582376 = balancing parens
- TM-57239683 = even number of 1s
- TM-3523796834721038296738259873 = Universal TM
- TM-3672349872381692309875823987609823712347823 = WindowsXP

Not the real numbers – they would be much much much much much bigger!

Acceptance Problem

Input: A Turing Machine M and an input w.

Output: Yes/no indicating if M eventually enters q_{Accept} on input w.

How can we state this as a language problem?

Acceptance Language

A_{TM} = { $<M, w>$ | M is a TM description and M accepts input w }

If we can decide if a string is in A_{TM}, then we can solve the Acceptance Problem (as defined on the previous slide).

Is A_{TM} ***Turing-recognizable***?

Turing-Recognizable

A language L is "Turing-recognizable" if there exists a TM M such that for all strings w:

- If $w \in L$ eventually M enters q_{accept}
- If $w \notin L$ either M enters q_{reject} **or** M never terminates

In a previous lecture, I incorrectly defined it as:

... if there exists a TM M such that for all strings w:

- If $w \in L$ eventually M enters q_{accept} ~~or M never terminates~~
- If $w \notin L$ either M enters q_{reject} **or** M never terminates

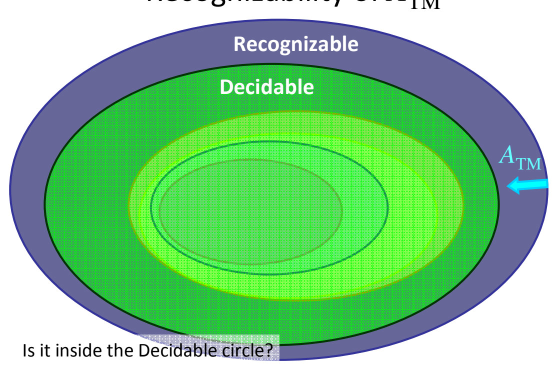
Why can't this be the right definition?

Recognizing A_{TM}

Run a UTM on $<M, w>$ to simulate running M on w. If the UTM accepts, $<M, w>$ is in A_{TM}.

M →, w → **Universal Turing Machine** → Output Tape for running TM-M starting on tape w

Recognizability of A_{TM}



Undecidability of A_{TM}

- Proof-by-contradiction. We will show how to construct a TM for which it is impossible to decide A_{TM}.

Assume there exists some TM H that **decides** A_{TM}.

Define D (<M>) = Construct a TM that:
Outputs the **opposite** of the result of simulating
H on input <M, <M>>

If M accepts its own description <M>, D(<M>) rejects.
If M rejects its own description <M>, D(<M>) accepts.

Reaching a Contradiction

Assume there exists some TM H that **decides** A_{TM}.

Define D (<M>) = Construct a TM that:
Outputs the **opposite** of the result of simulating
H on input <M, <M>>

If M accepts its own description <M>, D(<M>) rejects.
If M rejects its own description <M>, D(<M>) accepts.

What happens if we run D on its own description, <D>?

substituting D for M…

If D accepts its own description <D>, D(<D>) rejects.
If D rejects its own description <D>, D(<D>) accepts.

Reaching a Contradiction

Assume there exists some TM H that **decides** A_{TM}.

Define D (<M>) = Construct a TM that:
Outputs the **opposite** of the result of simulating
H on input <M, <M>>

If D **accepts** <D>:
H(D, <D>) **accepts** and D(<D>) **rejects**
If D **rejects** <D>,
H(D, <D>) **rejects** and D(<D>) **accepts**

Whatever D does, it must do the opposite, so there is a contraction!

Proving Undecidability

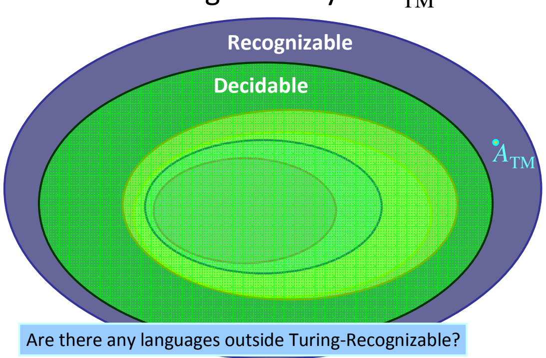
Assume there exists some TM H that **decides** A_{TM}.

Define D (<M>) = Construct a TM that:
Outputs the **opposite** of the result of simulating
H on input <M, <M>>

Whatever D does, it must do the opposite, so there is a contraction!

So, D **cannot** exist. But, if H exists, we know how to make D.
So, H **cannot** exist. Thus, there is no TM that decides A_{TM}.

Recognizability of A_{TM}



Are there any languages outside Turing-Recognizable?

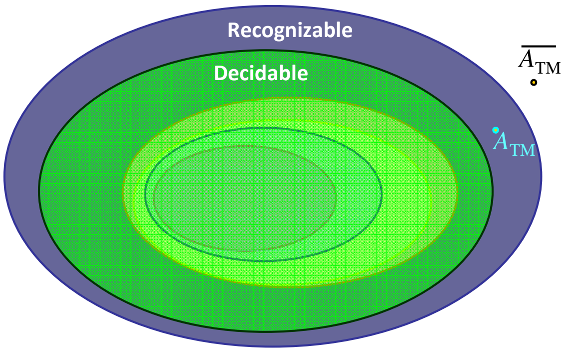
Recall: Turing-Recognizable

A language L is "Turing-recognizable" if there exists a TM M such that for all strings w:

- If $w \in L$ eventually M enters q_{accept}
- If $w \notin L$ either M enters q_{reject} **or** M never terminates

If M is Turing-recognizable and the complement of M is Turing-recognizable, what is M?

An Unrecognizable Language



Charge

- Next week:
  - How to you prove a problem is undecidable
  - How long can a TM that eventually halts run?
- PS5 will be posted by Saturday, and due **April 1** (this is a change from the original syllabus when it was due March 27)
- Exam 2 will be April 8 as originally scheduled