Undecidability

Lecture 11

Today

Decidable Problems (Languages)

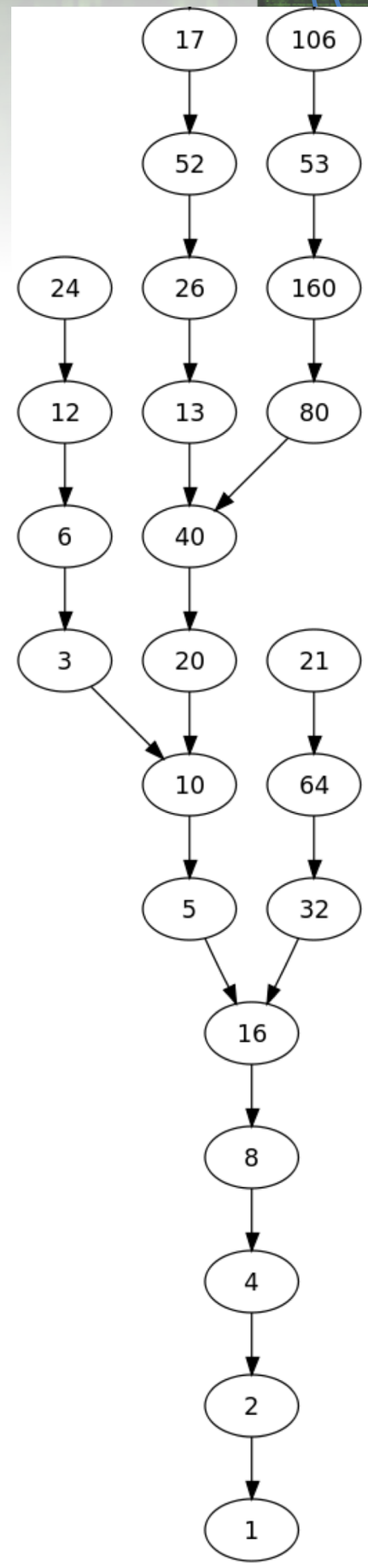
Undecidable Problems (Languages)

Proving undecidability

Using reductions to prove more undecidability

A Puzzle

```
Program Collatz (n:integer)
   while n > 1 {
     if  Even(n) then n := n/2
     else  n := 3n+1
   }
```



Q: Is there n that makes this program run forever?

Conjecture: `Collatz(n)` halts for every n > 0

We know that up to numbers with several hundred digits (10^{120}) it halts.

Can we write a program?

Could ask: isn't there a program that can help us decide if it is possible that is gets stuck in infinite loop?

There is no program that takes input an arbitrary piece of code and report correctly if it halts or not!

There is no automatic/systematic way to answer that question in general, but maybe that question has a particular answer.

Another Puzzle

Three stacks of cards, all cards in a stack identical. Two strings written on each card, one in green and one in red.

0	01	110
100	00	11

I want to choose a sequence of cards,
each from a different stack so that when
I line up the strings in green and red they are the same string.

Another Puzzle

There is solution!

0 100	01 00	110 11

110 11	01 00	110 11	0 100

I want to choose a sequence of cards,
each from a different stack so that when I line up the strings
in green and red they are the same string.

Another Puzzle

0 000	0 0101	10 1	1111 10

There is solution that uses 451 cards!

I want to choose a sequence of cards,
each from a different stack so that when I line up the strings
in green and red they are the same string.

Can we write a program?

Could ask: isn't there a program that can help us decide if there is solution?

Input is finite set of pairs of strings (2n)

Yes/No Question

There is no such algorithm!

Post correspondance problem has no solution!

How to show there is no algorithm?

- A TM on input a string w can : accept w, reject w, diverge on w (run forever)

Language of a TM T:

$ACCEPT(T) = \{ w \mid T \text{ accepts } w \}$

$REJECT(T) = \{ w \mid T \text{ rejects } w \}$

$HALT(T) = \{ w \mid T \text{ halts on } w \}$

$DIVERGE(T) = \{ w \mid T \text{ diverges on } w \}$

- We say T accepts L iff L=ACCEPT(T)
- What happens to the strings that are not accepted?
- We say T decides L iff L=ACCEPT(T) and DIVERGE(T)=ø

Decidability

- A language L is decidable iff some TM T decides L
- A language L is acceptable iff some TM T accepts L
- A language L is undecidable iff no TM T decides L
- A language L is unacceptable iff no TM T accepts L
  - TM decides a language = algorithm
  - if TM does not halt in some inputs its not really an algorithm
  - Getting the YES answers right is not enough!

No algorithm for a problem?

- There is no algorithm for a problem means the language associated with this problem is undecidable.

- E.g. Post correspondence problem: input is a string. Output is YES/NO. Language of all the instances where the answer is yes is a language L. No solution means L is not decidable.

- Same for the language L of programs that never enter an infinite loop. L is undecidable.

- Not hard to show that there are undecidable languages (uncountable number of languages, countable number of TM).

Example of Decidable Language

Acceptance Problem: Given a DFA, and a string w
Decide if the DFA accepts w.
Can be encoded as a language
A_{DFA}: {<B,W>| B is valid description of a DFA that accepts w}

Theorem: A_{DFA} is decidable
Proof:

Example of Decidable Language

Acceptance Problem: Given a DFA, and a string w
Decide if the DFA accepts w.
Can be encoded as a language
A_{DFA}: {<B,W>| B is valid description of a DFA that accepts w}

Theorem: A_{DFA} is decidable
Proof: Need to simply present a TM M that decides A_{DFA}.
M= “on input <B,w> simulate B on W.
If the simulation ends in accepting state, accept.
Otherwise reject.
-check if B is correctly describing a DFA
-keep track of current state of B and position in the input,
by writing it down on the tape
-update states and position according to transition function of B

Example of Decidable Language

Acceptance Problem: Given a NFA, and a string w
Decide if the NFA accepts w.
Can be encoded as a language
A_{NFA}: {<B,W>| B is valid description of a NFA that accepts w}

Idea:
Have TM N use previous TM M as subroutine.
Transform NFA to DFA first!

Example of Decidable Language

Acceptance Problem: Given a NFA, and a string w
Decide if the NFA accepts w.
Can be encoded as a language
A_{NFA}: {<B,W>| B is valid description of a NFA that accepts w}

Idea:

Example of Decidable Language

Acceptance Problem: Given aRegExp, and a string w
Decide if the The regular expression generates w.
Can be encoded as a language
A_{REX}: {<R,W>| R is valid description of a regular expression that generates w}

Idea:
Have TM use previous TM M as subroutine.
Transform Regular expression to NFA first!

Example of Decidable Language

Acceptance Problem: Given DFA A, decide if the language it accepts is empty

E_{DFA}={<A>| A is valid description of a DFA and L(A)=0}

Theorem: E_{DFA} is decidable.

Proof:

Example of Decidable Language

Acceptance Problem: Given DFA A, decide if the language it accepts is empty

E_{DFA}={<A>| A is valid description of a DFA and L(A)=0}

Theorem: E_{DFA} is decidable.

Proof: A DFA accepts some string if reaching an accept state from The start state by traveling along the arrows of the DFA is possible. Design a TM T that uses a marking algo to test this condition:

T= “on input <A>

-mark the start state of A

Repeat until no new states get marked: Mark any state that has a Transition coming into it from any state that is already marked.

-If no accept state is marked, accept. Otherwise reject.”

Example of Undecidable Language

$SELFREJECT$ = { <M> | M rejects <M> }

M =Turing Machine (piece of executable code)

<M> = encoding of M as a string (source code for M)

<M> is what you would feed to a universal TM,

that would allow it to simulate M.

(e.g. TM that rejects everything.

TM that rejects every description of a TM are in that language

Example of Undecidable Language

$SELFREJECT = \{ \langle M \rangle \mid M \text{ rejects } \langle M \rangle \}$

Claim: *SELFREJECT* is undecidable

Proof:

Suppose (towards contradiction) that there is a TM SR that decides SELFREJECT.

What happens if we feed <SR> to SR?

- SR accepts <SR>
- SR rejects <SR>
- SR diverges on <SR>

Example of Undecidable Language

$$SELFREJECT = \{ \langle M \rangle \mid M \text{ rejects } \langle M \rangle \}$$

Claim: *SELFREJECT* is undecidable

Proof:

Suppose (towards contradiction) that there is a TM SR that decides SELFREJECT.

What happens if we feed <SR> to SR?

- SR accepts <SR> :
- SR rejects <SR>
- SR diverges on <SR> :

Example of Undecidable Language

$$SELFREJECT = \{ \langle M \rangle \mid M \text{ rejects } \langle M \rangle \}$$

Claim: *SELFREJECT* is undecidable

Proof:

Suppose (towards contradiction) that there is a TM SR that decides SELFREJECT.

What happens if we feed <SR> to SR?

- SR accepts <SR> : <SR> is in SELFREJECT = SR rejects <SR>!
- SR rejects <SR> : <SR> not in SELFREJECT = SR accepts or diverges
- SR diverges on <SR> : impossible because this TM makes decisions about every string!

Russel's barbers paradox

- On a small town, on a certain day everyone gets a haircut

- Everyone either cuts their own hair or has their hair cut by someone else.

- Barber = cuts the hair only of those who don't cut their own

- Does the barber cut their own hair?

Russel's barbers paradox

- On a small town, on a certain day everyone gets a haircut

- Everyone either cuts their own hair or has their hair cut by someone else.

Every TM A cuts B's hair = A accepts <B>

- Barber = cuts the hair only of those who don't cut their own

Barber is TM that accepts <M> iff M does not accept <M>

- Does the barber cut their own hair?

Cantor

Let X be any set and 2^X its powerset (Set of subsets)

f: X $\rightarrow 2^X$ is NOT onto

Meaning there is a set S in 2^X that has no preimage.

- x is SAD if x is not in f(x)
- Y = {x in X | x is SAD}
- There is no y in X such that f(y)= Y

Cantor

Let X be any set and 2^X its powerset (Set of subsets)

f: X $\rightarrow 2^X$ is NOT onto

Meaning there is a set S in 2^X that has no preimage.

- x is SAD if x is not in f(x)
- Y = {x in X | x is SAD}
- There is no y in X such that f(y)= Y
- maps to barber paradox (X is the set of people, f(x) is the set of people whose hair x cuts). A person is sad if they don't cut their own hair
- maps to SELFREJECT : X is the set of TM that halt on all inputs. f(x) is the set of all TM that x accepts. A TM is sad if is rejects its own encoding

Cantor's Diagonal Slash

Is the set of all infinitely long binary strings countable?

Suppose it was: consider *enumerating* them in a table

Consider the string corresponding to the "flipped diagonal"

It doesn't appear in this table!

0	1	0	0	1	1	1	.	.

S_i								
S_1 =	1	0	0	1	0	0	0	0	1
S_2 =	0	0	1	0	1	0	0	1	1
S_3 =	1	1	1	1	1	1	1	0	0
S_4 =	1	1	0	1	0	1	0	1	1
S_5 =	1	1	0	0	0	0	1	0	0
S_6 =	0	0	0	0	0	0	1	1	0

Showing Undecidability

To show L is undecidable, reduce some undecidable language to

$$SELFHALT = \{ \langle M \rangle \mid M \text{ halts on } \langle M \rangle \}$$

Claim: *SELFHALT* is undecidable

More general looking problem:

$$HALT = \{ \langle M, w \rangle \mid M \text{ halts on } w \}$$

Claim: *HALT* is acceptable

The halting problem

Claim: *HALT* is undecidable