COMP30026 Models of Computation

Decibale and Undecibale Problems

Lecture Week 11. Par

Semester 2, 2021

Alan Turing

Alan Turing was born in 1912. At that time, "computer" was a job title: a human employed to do tedious numeric

Legacy: "Turing machine", the "Church-Turing thesis", "Turing reduction", the "Turing test", the "Turing awar more.

One of Turing's great accomplishments was to pu ity" on a firm foundation and to establish that certain important problems do not have an algorithmic solution.

We Have Many Models of Computability

Turing machines (A. Turing, 1936)

Lambda calculus (A. Church, 1936)

Partial re

Post sys

Markov algorithms (A. Markov, 1954)

While programs

Register machines

Horn clauses

Post Markov

⋮

The Church-Turing Thesis

The class of computable functions is exactly the class of functions that can be realised by

External evidence: All the above models ar
of the fact that they all look very differe
independently.

Internal evidence: It seems that no matter how we "extend" any of them, we fail to get something that is more powerful.

Decidable Problems

We can phrase these problems as language decidability problems.

For example, the acceptance problem for DFAs is whether, given a DFA D

Since we c m can be seen as testing for membership of the languag

$$A_{DFA} = \{\langle D, w\rangle \mid D \text{ is a DFA th}$$

By $\langle D, w\rangle$ we mean a (string) encoding of the pair D, w.

DFA Acceptance Is Decidable

Theorem: A_{DFA} is a decidable language.

Proof sketch: The crucial point is that it is possible for a Turing machine M to simulate a DFA D.

M finds o

$\underbrace{1 \ldots n}_{Q}$ δ $\# \underbrace{baa \ldots}_{w} \$$

First M checks that the first five components rep
and if not, rejects.

Then M simulates the moves of D, keeping track of D's state and the current position in w, by writing these details on its tape, after \$.

When the last symbol in w has been processed, M accepts if D is in a state in F, and rejects otherwise.

TMs as Interpreters

We won't give the details of how the Turing machine simulates the DFA. Many tedious low-level programming steps are involved.

However, it should be clear that it is possible for a Turing machine to mimic D

The desc

that a Turing machine can act as an interpreter for thi ge.

Turing machines themselves can be encoded as str en a Turing machine can interpret Turing machines

This is no more strange than the fact that we can write an interpreter for Haskell, say, in Haskell.

Theorem:

$$A_{NFA} = \{\langle N, w\rangle \mid N \text{ is an NFA that accepts } w\}$$

is a decida

Proof sk

equivalent DFA was mechanistic and terminati uring machine can do that job.

Having written the encoding of the DFA on its tape, the Turing machine can then "run" the machine M from the previous proof.

DFA Equivalence Is Decidable

Theorem:

$$EQ_{DFA} = \{\langle A, B\rangle \mid A \text{ and } B \text{ are DFAs and } L(A) = L(B)\}$$

is decidable.

Proof sk

from DF

These procedures are mechanistic and finite—a halting Turing machine M can perform them.

Hence from A and B, M can produce a DF

$$L(C) = \big(L(A) \cap L(B)^c\big) \cup \big(L(A)^c \cap L(B)\big)$$

Note that $L(C) = \emptyset$ iff $L(A) = L(B)$.

So M just needs to use the emptiness checker on C.

Generation by CFGs Is Decidable

Theorem:

$$A_{CFG} = \{\langle G, w\rangle \mid G \text{ is a CFG that generates } w\}$$

is decida

The proo

particular equivalent form, Chomsky Normal Form.

In Chomsky Normal Form, each production take s:

$A \to B\ C$ or

(With one exception:
We also allow $S \to \epsilon$, where S is the grammar's start variable.)

Generation by CFGs Is Decidable

For every *w* can be derived t

So to decid *CFG*
that length, in finite time, and see if one generates

Every CFL Is Decidable

Two slides back we saw that it is decidable whether a CFG G generates a string w.

The deci

Now we

Theorem: Every context-free language

Proof: This is just saying that we can .
Let G_0 be a CFG for L_0. The decider for L_0 simply takes input w and runs S on $\langle G_0, w \rangle$.

Every CSL Is Decidable

Theorem: For every context-sensitive language L there is a linear bounded automaton (TM with a bounded tape) M, such that M recognises L.

Theore) is decidab

Proof: h n is at most $|Q| \cdot n \cdot |\Gamma|^n$, where $|Q|$ is the num $|$ is the size of the tape alphabet (the tape has at most

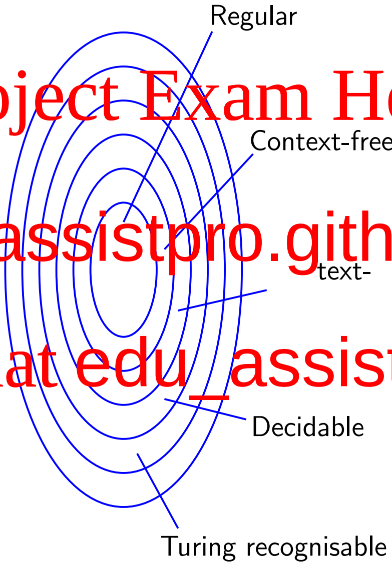
If M accepts w of length n then M does so within at most $|Q| \cdot n \cdot |\Gamma|^n$ steps. Any computation of length more than $|Q| \cdot n \cdot |\Gamma|^n$ is “cycling” and so cannot accept w. If M can't accept w within $|Q| \cdot n \cdot |\Gamma|^n$ steps, it rejects this string.

The Hierarchy of Language Classes

The diagram shows the relations amongst language classes esta

But are ther recognisable languages that are not decidable?

As it turns out, yes.



Assignment Project Exam Help

https://eduassistpro.github.i

Add WeChat edu_assist_pr

An Undecidable Language

Now let us study undecidable problems/languages.

We start by showing that it is undecidable whether a Turing machine accepts a g

$\}$

is undecidable.

The main difference from the case of A

Turing machine may fail to halt.

TM Acceptance Is Undecidable

Theorem:

$$A_{TM} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ accepts } w\}$$

is undecidable.

Proof: M H:

H M, w *reject* if M

Using H we can construct a Turing machine whether a given machine M fails to accept i $M\rangle$:

1. Input is $\langle M\rangle$, where M is some Turing machine.
2. Run H on $\langle M, \langle M\rangle\rangle$.
3. If H accepts, reject. If H rejects, accept.

TM Acceptance

In summary:

$$D(\langle M \rangle) = \begin{cases} \textit{accept} & \text{if } M \text{ does not accept } \langle M \rangle \\ \textit{reject} & \text{if } M \text{ accepts } \langle M \rangle \end{cases}$$

But no ma

Why? Because we obtain an absurdity when we inve D s behaviour when we run it on its own encoding:

$$D(\langle D \rangle) = \begin{cases} \textit{accept} & \text{if } D \text{ d} \\ \textit{reject} & \text{if } D \text{ accepts } \langle D \rangle \end{cases}$$

Hence neither D nor H can exist.

Comparing Sizes of Sets: Cantor's Criterion

So what does 'equals' and 'less' mean for infinite cardinality?

How do we

Cantor'

- $card(X) \leq card(Y)$ iff there is a t Y.
- $card(X) = card(Y)$ iff
  $card(X) \leq car$ $rd(X)$.

As a consequence, there are (infinitely) many degrees of infinity.

X is countable iff $card(X) \leq card(\mathbb{N})$.

X is countably infinite iff $card(X) = card(\mathbb{N})$.

Exampl
number

Importantly, Σ^* is countable for all finite alpha
alphabet of printable characters on your keyboar

$\mathcal{P}(\mathbb{N})$, $\mathbb{N} \to \mathbb{N}$, and $\mathbb{Z} \to \mathbb{Z}$ are uncountable, as can be shown by diagonalisation.

Diagonalisation Showing $\mathbb{Z} \to \mathbb{Z}$ Is Uncountable

Theorem: There is no bijection $h : \mathbb{N} \to (\mathbb{Z} \to \mathbb{Z})$.

Proof: Assume h exists. Then

contain

Now construct $f : \mathbb{Z} \to \mathbb{Z}$ as follows:

$$f(n) = h(n)(n$$

Then $f \neq h(n)$ for all n, so we have a contradiction.

Why This Is Called Diagonalisation

Here is some hypothetical listing of all the functions $h(0), h(1), \ldots$ that make up $\mathbb{Z} \to \mathbb{Z}$:

	0	1	2	3	4	5	…
h(3)	6	93	17	84	6	93	…
h(4)	45	18	-8	-5	63	-9	…
⋮							

Why This Is Called Diagonalisation

Here is some hypothetical listing of all the functions $h(0), h(1), \ldots$ that make up $\mathbb{Z} \to \mathbb{Z}$:

	0	1	2	3	4	5	...
h(3)	6	93	17				
h(4)	45	18	-8	-5			
⋮							

f is defined in such a way that it cannot possibly be in the listing:

	0	1	2	3	4	5	...
f	20	43	45	85	64	...	...

Algorithms vs Functions

Consider the set of algorithms that realise functions $f : \mathbb{Z} \to \mathbb{Z}$.

How large is that set?

It is infinit *, where Σ is the se et is countable.

So there cannot be any more, say, Has `Integer -> Integer` than there are intege function is represented finitely, as a finite sequence of symbols from a finite alphabet.

Algorithms vs Functions

However, we saw that $\mathbb{Z} \to \mathbb{Z}$ is not countable.

In other w ct, lots of them) th

So are there any "important" functions that are not c le?

As it turns out, yes, very much so!

Problems that Have No Algorithmic Solution

Some undecidable problems:

- Ar
- Ar
- Is a gi
- Will a given Python program halt for all input?
- Will a given Java program ever throw a certain e

Next week we will explore some other undecidable problems.