CSE 2001:
Introduction to Theory of Computation

Fall 2012

Suprakash Datta

datta@cse.yorku.ca

Office: CSEB 3043

Phone: 416-736-2100 ext 77875

Course page: http://www.cs.yorku.ca/course/2001

Next: Ch 4.2

Towards undecidability:

- **The Halting Problem**
- **Diagonalization arguments**

The Halting Problem

The existence of the universal TM U shows that
$A_{TM} = \{\langle M,w \rangle \mid M \text{ is a TM that accepts } w\}$
is TM-recognizable, but can we also *decide* it?

The problem lies with the cases when M does not halt on w. In short: the halting problem.

We will see that this is an insurmountable problem: in general one cannot decide if a TM will halt on w or not, hence A_{TM} is undecidable.

Counting arguments

- We need tools to reason about undecidability.
- The basic argument is that there are more languages than Turing machines and so there are languages than Turing machines. Thus some languages cannot be decidable.

Countable sets in language theory

- Σ^* is countable – finitely many strings of length k. Order them lexicographically.
- Set of all Turing machines countable – every TM can be encoded as a string over some Σ.

Uncountable Sets

There are infinite sets that are not countable.
Typical examples are R, $\mathcal{P}$(N) and $\mathcal{P}$({0,1}*)

We prove this by a diagonalization argument.
In short, if S is countable, then you can make a list $s_1,s_2,\ldots$ of all elements of S.

Diagonalization shows that given such a list, there will always be an element x of S that does not occur in $s_1,s_2,\ldots$

Uncountability of $\mathcal{P}$(N)

The set $\mathcal{P}$(N) contains all the subsets of $\{1,2,\ldots\}$.

Each subset $X \subseteq N$ can be identified by an infinite string of bits $x_1x_2\ldots$ such that $x_j=1$ iff $j \in X$.

There is a bijection between $\mathcal{P}$(N) and $\{0,1\}^N$.

Proof by contradiction: Assume $\mathcal{P}$(N) countable.

Hence there must exist a surjection F from N to the set of infinite bit strings.

"There is a list of *all* infinite bit strings."

Diagonalization

Try to list all possible infinite bit strings:

0	0	0	0	0	0	…
1	1	1	1	1	1	…
2	1	0	0	0	0	…
3	0	1	0	1	0	…
⋮						⋱

Look at the bit string on the diagonal of this table: 0101… The negation of this string (“1010…”) does not appear in the table.

No Surjection $\mathbb{N} \to \{0,1\}^{\mathbb{N}}$

Let F be a function $N \to \{0,1\}^N$.
F(1),F(2),… are all infinite bit strings.

Define the infinite string $Y=Y_1Y_2\ldots$ by
Y_j = **NOT**(j-th bit of F(j))

On the one hand $Y \in \{0,1\}^N$, but on the other hand: for every $j \in N$ we know that $F(j) \neq Y$ because F(j) and Y differ in the j-th bit.

F cannot be a surjection: $\{0,1\}^N$ is uncountable.

Generalization

- We proved that $\mathcal{P}(\{0,1\}^*)$ is uncountably infinite.
- Can be generalized to $\mathcal{P}(\Sigma^*)$ for any finite Σ.
- Can be used to show that the set of reals is uncountable (last class).

Uncountability

We just showed that there it is impossible to have a surjection from N to the set $\{0,1\}^N$.

What does this have to do with Turing machine computability?

Counting TMs

Observation: Every TM has a finite description; there is only a countable number of different TMs. (A description $\langle M \rangle$ can consist of a finite string of bits, and the set $\{0,1\}^*$ is countable.)

Our definition of Turing recognizable languages is a mapping between the set of TMs $\{M_1, M_2, \ldots\}$ and the set of languages $\{L(M_1), L(M_2), \ldots\} \subseteq \mathcal{P}(\Sigma^*)$.

Question: How many languages are there?

Counting Languages

There are uncountably many different languages over the alphabet Σ={0,1} (the languages $L \subseteq \{0,1\}^*$).
With the lexicographical ordering ε,0,1,00,01,… of Σ^*, every L coincides with an infinite bit string via its <u>characteristic sequence</u> χ_L.

Example for L={0,00,01,000,001,…} with χ_L= 0101100…

Σ^*	ε	0	1	00	01	10	11	000	001	010	⋯
L		X		X	X			X	X	X	⋯
χ_L	0	1	0	1	1	0	0	1	1	1	⋯

Counting TMs and Languages

There is a bijection between the set of languages over the alphabet $\Sigma=\{0,1\}$ and the uncountable set of infinite bit strings $\{0,1\}^N$.

- There are uncountable many different languages $L\subseteq\{0,1\}^*$.
- Hence there is no surjection possible from the countable set of TMs to the set of languages. Specifically, the mapping L(M) is not surjective.

Conclusion: There are languages that are not Turing-recognizable. (A lot of them.)

Is This Really Interesting?

We now know that there are languages that are not Turing recognizable, but we do not know what kind of languages are non-TM-recognizable.

Are there interesting languages for which we can prove that there is no Turing machine that recognizes it?

Proving Undecidability (1)

Recall the language

A_{TM} = { <M,w> | M is a TM that accepts w }.

Proof that A_{TM} is not TM-decidable (Thm. 4.11)

(Contradiction) Assume that TM G decides A_{TM}:

$$G\langle M, w\rangle = \begin{cases} \text{"accept" if M accepts w} \\ \text{"reject" if M does not accept w} \end{cases}$$

From G we construct a new TM D that will get us into trouble…

Proving Undecidability (2)

The TM D works as follows on input $\langle M \rangle$ (a TM):

1) Run G on $\langle M, \langle M \rangle\rangle$
2) Disagree with the answer of G

(The TM D always halts because G always halts.)

In short: $$D\langle M \rangle = \begin{cases} \text{"accept" if G rejects } \langle M, \langle M \rangle \rangle \\ \text{"reject" if G accepts } \langle M, \langle M \rangle \rangle \end{cases}$$

Hence: $$D\langle M \rangle = \begin{cases} \text{"accept" if M does not accept } \langle M \rangle \\ \text{"reject" if M does accept } \langle M \rangle \end{cases}$$

Now run D on $\langle D \rangle$ ("on itself")…

Proving Undecidability (3)

Result: $$D\langle D\rangle = \begin{cases} \text{"accept" if D does not accept } \langle D\rangle \\ \text{"reject" if D does accept } \langle D\rangle \end{cases}$$

This does not make sense: D only accepts if it rejects, and vice versa.
(Note again that D always halts.)

Contradiction: **A_{TM} is not TM-decidable**.

This proof used diagonalization implicitly…

Review of Proof (1)

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\cdots$
M_1	accept		accept		
M_2	accept	accept	accept	accept	
M_3					$\cdots$
M_4	accept	accept			
$\vdots$			$\vdots$		$\ddots$

'Acceptance behavior' of M_i on $\langle M_j \rangle$

Review of Proof (2)

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\cdots$
M_1	accept	reject	accept	reject	
M_2	accept	accept	accept	accept	
M_3	reject	reject	reject	reject	$\cdots$
M_4	accept	accept	reject	reject	
$\vdots$			$\vdots$		$\ddots$

'Deciding behavior' of G on <M_i,<M_j>>

Review of Proof (3)

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\cdots$	$\langle D \rangle$	$\cdots$
M_1	accept	reject	accept	reject			
M_2	accept	accept	accept	accept			
M_3	reject	reject	reject	reject	$\ldots$		
M_4	accept	accept	reject	reject			
$\cdot$			$\cdot$		$\cdot$		

Review of Proof (4)

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\cdots$	$\langle D \rangle$	$\cdots$
M_1	accept	reject	accept	reject			
M_2	accept	accept	accept	accept			
M_3	reject	reject	reject	reject	$\ldots$		
M_4	accept	accept	reject	reject			
$\vdots$			$\vdots$		$\ddots$		
D	reject	reject	accept	accept	$\cdots$	?	
$\vdots$							

Contradiction for D on input <D>.

TM-Unrecognizable

A_{TM} is not TM-decidable, but it is TM-recognizable. What about a language that is not recognizable?

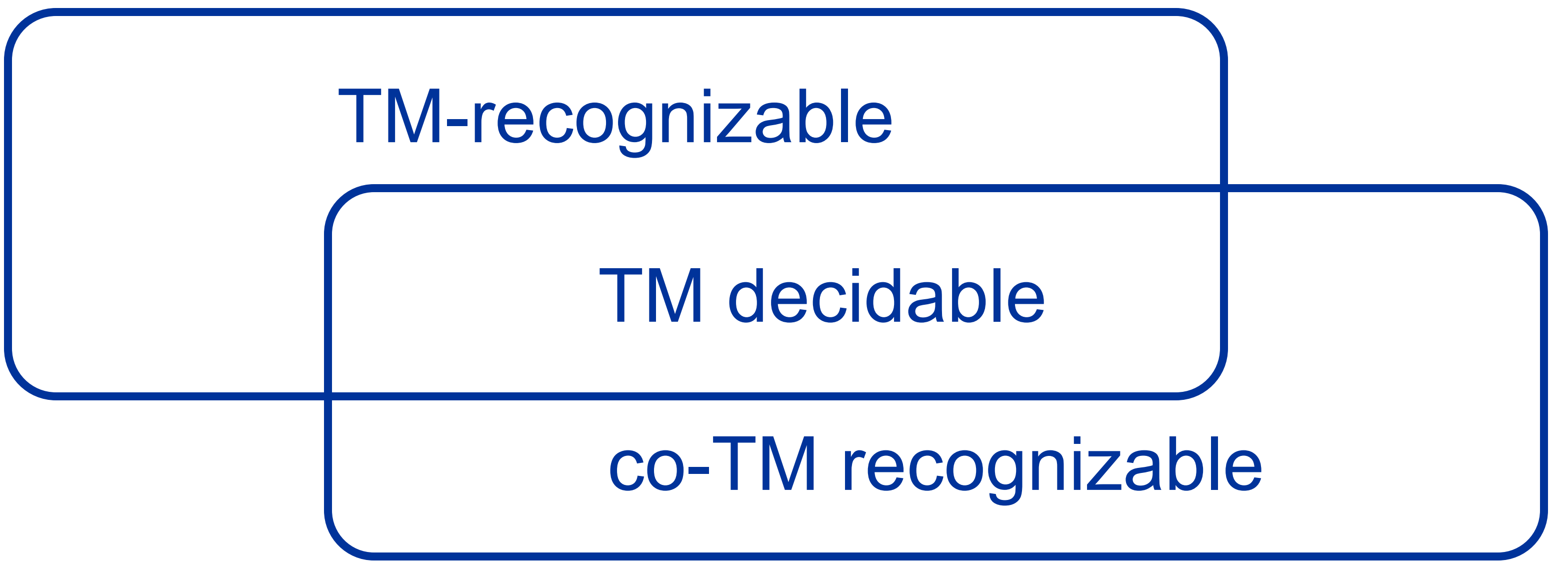
Theorem 4.22: If a language A is recognizable and its complement $\bar{A}$ is recognizable, then A is Turing machine decidable.

Proof: Run the recognizing TMs for A and $\bar{A}$ in parallel on input x. Wait for one of the TMs to accept. If the TM for A accepted: "accept x"; if the TM for $\bar{A}$ accepted: "reject x".

$\bar{A}_{TM}$ is not TM-Recognizable

By the previous theorem it follows that $\bar{A}_{TM}$ cannot be TM-recognizable, because this would imply that A_{TM} is TM decidable (Corollary 4.23).

We call languages like $\bar{A}_{TM}$ <u>co-TM recognizable</u>.



Things that TMs Cannot Do:

The following languages are also unrecognizable:

$E_{TM} = \{ \langle G \rangle \mid G \text{ is a TM with } L(G)=\varnothing \}$

$EQ_{TM} = \{ \langle G,H \rangle \mid G \text{ and } H \text{ are TMs with } L(G)=L(H) \}$

To be precise:

- E_{TM} is co-TM recognizable
- EQ_{TM} is not even co-Turing recognizable

How can we prove these facts?

Next: reducibility

- We still need to ***prove*** that the Halting problem is undecidable.
- Do more examples of undecidable problems.
- Try to get a general technique for proving undecidability.

Halting problem

- Assume that it is decidable. So there is a TM S that decides

HALT={<M,w>|M is a TM and M halts on w}

- Use S as a subroutine to get a TM S to decide

A_{TM} = {<M,w> | M is a TM that accepts w }

- Therefore A_{TM} is decidable. CONTRADICTION!
- Details follow ….

Halting problem - 2

S = “On input <M,w>

- Run TM R on input <M,w>.
- If R rejects, REJECT.
- If R accepts, simulate M on w until it halts.
- If M has accepted, ACCEPT, else REJECT”

More undecidability

E_{TM} = {<M>| M is a TM and L(M) = ϕ}
We mentioned that E_{TM} is co-TM recognizable.
We will prove next that E_{TM} is undecidable.

Intuition: You cannot solve this problem UNLESS you solve the halting problem!!

But this is hard to formalize, so we use A_{TM}. Instead.

E_{TM} is undecidable

Assume R decides E_{TM}. Use R to design TM S to decide A_{TM}.

- Given a TM M and input w, define a new TM M':
  - If $x \neq w$, reject
  - If $x = w$, accept iff M accepts w

S = "On input <M,w>

- Construct M' as above.
- Run TM R on input <M'>.
- If R accepts, REJECT; If R rejects, ACCEPT."