Lecture 6

The Acceptance Problem for TMs

A_{TM} = { <M,w> | M is a TM & w $\in$ L(M) }

Theorem: A_{TM} is Turing recognizable

Pf: It is recognized by a TM U that, on input <M,w>, simulates M on w step by step. U accepts iff M does. □

U is called a *Universal Turing Machine*

(Ancestor of the stored-program computer)

Note that U is a recognizer, not a decider.

A_{TM} is Undecidable

A_{TM} = { <M,w> | M is a TM & w $\in$ L(M) }

Suppose it's decidable, say by TM H. Build a new TM D:

"on input <M> (a TM), run H on <M,<M>>; when it halts, halt & do the opposite, i.e. accept if H rejects and vice versa"

D accepts <M> iff H rejects <M,<M>> (by construction)
iff M rejects <M> (H recognizes A_{TM})

D accepts <D> iff D rejects <D> (special case)

Contradiction!

A *specific* non-Turing-recognizable language

Let M_i be the TM encoded by w_i, i.e. $\langle M_i \rangle = w_i$

(M_i = some default machine if w_i is an illegal code.)

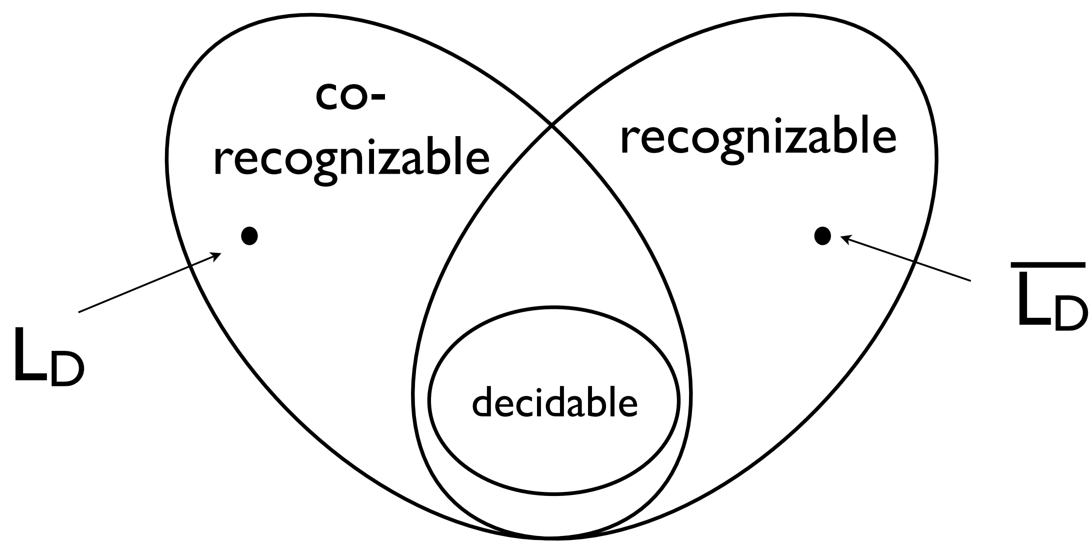
i, j entry tells whether M_i accepts w_j

Then L_D is *not recognized by any TM*

	w_1	w_2	w_3	w_4	w_5	w_6	
$\langle M_1 \rangle$	0	0	0	0	0	0	
$\langle M_2 \rangle$	1	1	1	1	1	1	
$\langle M_3 \rangle$	0	1	0	1	0	1	
$\langle M_4 \rangle$	0	1	0	0	0	0	...
$\langle M_5 \rangle$	1	1	1	0	0	0	
$\langle M_6 \rangle$	0	1	0	0	0	1	
			⋮				⋱
L_D	1	0	1	1	1	0	...

Note: The above TM D, if it existed, would recognize exactly the language L_D defined in this diagonalization proof (which we already know is not recognizable)

Decidable $\subsetneq$ Recognizable

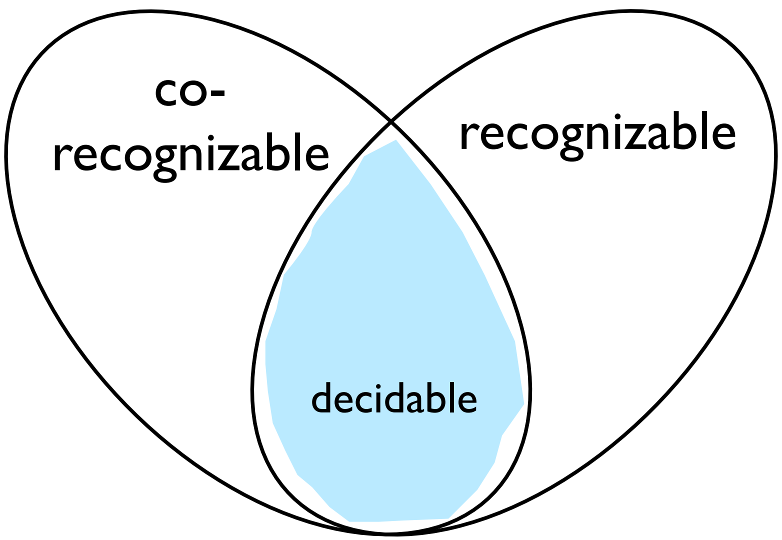


Decidable = Rec ∩ co-Rec

L decidable iff both L & L^c are recognizable

Pf:
(⇐) on any given input, dovetail a recognizer for L with one for L^c; one or the other must halt & accept, so you can halt & accept/reject appropriately.

(⇒): from last lecture, decidable languages are closed under complement (flip acc/rej)



Reduction

“A is reducible to B” means I could solve A *if* I had a subroutine for B

Ex:

Finding the max element in a list is reducible to sorting

pf: sort the list in increasing order, take the last element

(A big hammer for a small problem, but never mind...)

The Halting Problem

$HALT_{TM}$ = { <M,W> | TM M halts on input w }

Theorem: The halting problem is undecidable

Proof:

$A = A_{TM}$, $B = HALT_{TM}$ Suppose I can reduce A to B. We already know A is undecidable, so must be that B is, too.

Suppose TM R decides $HALT_{TM}$. Consider S:

> On input <M,w>, run R on it. If it rejects, halt & reject; if it accepts, run M on w; accept/reject as it does.

Then S decides A_{TM}, which is impossible. R can't exist.