CPSC 421/501 Oct. 19

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

=

$L_{visit\_state} = \{ \langle M, w, q \rangle \mid \text{on input } w,\ M \text{ reaches } q \}$

(equivalent of dead code)

=

$L_{dead\_code\_in\_C}$

$= \{ \langle p, i \rangle \mid p \text{ is a C-program}, i \text{ is an input to C-prog}, p \text{ has at least one line that is never visited} \}$

=

. —

A_{TM} can be recognized, but not decided.

Why?

Recognized: given $\langle M, w \rangle$, we can simulate M on input w by some fixed machine, a "Universal Turing Machine"

Can't be decided: If H decided A_{TM}, then get contradiction:

Consider D takes input w, if $w = \langle M \rangle$ then D

~~outputs~~ accepts $w = \langle M \rangle$ if $\langle M, \langle M \rangle \rangle \notin A_{TM}$ and reject $w = \langle M \rangle$ if $\langle M, \langle M \rangle \rangle \in A_{TM}$.

=

Consider D on input $\langle D \rangle$:

D decides: accepts or rejects

(in some finite time)

If D accepts $\langle D \rangle$, then

$\langle D, \langle D \rangle \rangle \notin A_{TM} \Rightarrow$ D does not accept $\langle D \rangle$

If D rejects $\langle D \rangle$ then

$\langle D, \langle D \rangle \rangle \in A_{TM}$

$\Rightarrow$ D accepts $\langle D \rangle$.

(Contradiction.)

=

"Turing machines can speak about themselves"

$w = \langle D \rangle \rightsquigarrow \langle D, \langle D \rangle \rangle$

description of D — string expresses D

$\bar{L} = \Sigma^* \setminus L$

$= \{ w \in \Sigma^* \mid w \notin L \}$

then if L and $\bar{L}$ can be recognized, then they both can be decided.

Why? $w \in \Sigma^*$

Is w in L or in $\bar{L}$

M_1: M_1 takes w as input, and ~~accepts~~ accepts w in finite time if $w \in L$

Since A_{TM} can be recognized but not decided,

$\overline{A_{TM}}$ = complement of A_{TM}

$= \{ w \in \Sigma^* \mid w \text{ is not in } A_{TM} \}$

cannot be recognized.

Proof: If A_{TM} and $\overline{A_{TM}}$ can be recognized, then they can both be decided...

Generally: If $L \subset \Sigma^*$ and

Otherwise: If we have $L_1, L_2 \subset \Sigma^*$, more generally

$L_1, L_2, L_3, L_4, L_5 \subset \Sigma^*$ ← $L_i \leftrightarrow$ machine M_i

run step 1 of M_1

run step 1 of M_2

⋮

run step 1 of M_5

run step 2 of M_1

-- step 2 of M_2

⋮

step 2 M_5

run step 3 of M_1

⋮

M_2 that if $w \notin L$, i.e. $w \in \bar{L}$

step 1

step 2

⋮

then M_2 in a finite amount of time accepts w.

Given w: run

one step M_1 → one step M_2

2nd step M_1 → 2nd step M_2

3rd step M_1 → 3rd step M_2

⋮ finite time ⋮ finite

If $w \in L$ If $w \in \bar{L}$

Then L can be decided in poly time.

If L can be ~~decided~~ recognized by M_1

$\overline{L}$ can be ~~decided~~ recognized by M_2

If

$w \in L$, M_1 accepts

M_2 never accepts

$w \notin L$ M_1 never accept

M_2 accept

then $L, \overline{L}$ can be decided

~~A_{TM} recognized~~

You can run a bunch of algorithms "in parallel" on one machine.

Thm: If $M_1, \ldots, M_5$ s.t. on any input, w, then if

$w \in L \Rightarrow$ one of $M_1, \ldots, M_5$ accepts w in poly time

and

$w \notin L \Rightarrow$ one of $M_1, \ldots, M_5$ reject w in polytime

and if

$w \in L \Rightarrow$ no machine $M_1, \ldots, M_5$ rejects w

$w \notin L \Rightarrow$ — — — — accepts w

DEAD-TM-STATE

$= \{ \langle M \rangle \mid$ M is a Turing machine with some state that is never reached on any input $\}$

Is there an algorithm that when given $\langle M \rangle$ with some state that is never reached it will accept (finite time).

UNDEAD-TM-STATE $= \{ \langle M \rangle$ s.t. M is a T.m each of whose states is reached $\}$

If L can be recognized, but not decided, then $\overline{L}$ cannot be recognized.

Pf: If not, $L, \overline{L}$ recognized,

$\Rightarrow L, \overline{L}$ can be decided.

DEAD-C-CODE $= \{ \langle p \rangle$ s.t. p is a C-program, p has a line of code that is never reached by any input $\}$