Last lecture: Reduction to $\text{NE}_{\text{TM}}$ can also be from $K_m$.

$K_m = \{ e : \text{Me accepts } e \}$ (in Gödel Number notation)

$K_m \leq_m \text{ALL}_{\text{TM}}$

$f(e) = e'$ where $e'$ is

the code of

the TM $Me'$ :=

{Simulate $Me$ on $e'$}

if and only if $e$ is accepted by $Me$.  

\[ \text{if input } x \text{ yields a halting state } e ' \text{ in } Me ' \]

\[ \text{then } f(e) = e ' \]

\[ \text{else } f(e) = \text{accept} x \]

\[ \text{end if} \]

\[ \text{end function} \]

We also get $eK \subseteq \Sigma^*$, so $K_m \leq_m \text{ALL}_{\text{TM}}$ by the same $f$.

\[ \text{End of proof} \]

$g(e) = e''$ code of $Me''$ :=

\[ eK \Rightarrow \text{Me accepts } e \Rightarrow \text{there is an } n \geq 1 \text{ such that } Me \text{ accepts } e \text{ in } n \text{ steps} \Rightarrow \]

\[ \exists x_1 \leq n_0 \text{ such that } Me'' \text{ sees this and rejects } x \]

\[ \Rightarrow L(Me'') \text{ is finite } \Rightarrow L(Me'') \neq \Sigma^* \]

\[ eK \Rightarrow \forall x \text{ Me'' never sees } Me(e) \text{ accept } \Rightarrow \forall x \text{ Me'' accept } x \Rightarrow L(Me'') = \Sigma^* \]

Thus $e \in \text{D}_{\text{TM}} \Rightarrow L(Me'') = \Sigma^*$, so $\text{D}_{\text{TM}} \leq_m \text{ALL}_{\text{TM}}$ via this $g$.

$g$ is also a computable "flowchart assembly" function.
Neither RE nor Co-RE.

ALLm has "More than one Degree of Undecidability."

Thus the language $\text{ALL}_m = \sum e \in L(m) \neq \Sigma^*$ is neither (e. nor (co-e. text style
i.e. neither if nor is complement is Turing recognizable.

Both switches can be applied with further "Language Filtering"

\[ f'(x, y) \rightarrow Me' = \begin{cases} 
\text{Sim } Me(x) & \text{if } e \in L Me \text{ accepts } \sum = \{0, 1\}^* \\
\text{Reject } x & \text{if } e \in L Me \text{ rejects } \sum = \{0, 1\}^* 
\end{cases} \]

which is regular, so $f(1) \notin \text{REGULAR}_m$.

\[ e \in L Me' \text{ is non-regular.} \]

We get $e \in D \Rightarrow f(e) \in \text{REGULAR}_m$ so $\text{REGULAR}_m$ is not (e.

\[ g' \xrightarrow{e}{c} Me'' \]

\[ e \in L Me'' \text{ is finite} \Rightarrow \text{f. is still checking.} \]

\[ f'(x, y) \rightarrow Me'' = \begin{cases} 
\text{Sim } Me(x) \text{ for } n \text{ steps} & \text{did it accept?} \\
\text{Reject } x & \text{yes} \\
\text{Accept } x \text{ if } x \in \{0, 1\}^* & \text{no}
\end{cases} \]

Thus $\text{REGULAR}_m$ is neither RE nor Co-RE. [EX: it has 3 degrees of undecidability]

This adds to what the text says in §5.1 about it. Study Cite: What happens if we put the $x \in \{0, 1\}^*$ test?

Note: $\text{REGULAR}_m$ is not the same as your HW (3) language:

\[ \text{REGULAR} = \{ L(G) = 6 \text{ is a CFG and } L(G) \text{ is regular?} \]
There are computable functions $h$ and $h'$ such that for any TM $M_e$:

- $h(e)$ is a CFG $G$ such that $L(G) = \Sigma^* \iff M_e \in \mathcal{E}_{TM} \implies L(M_e) = \emptyset$ and such that if $M_e \in \mathcal{E}_{TM}$, $L(G)$ is not regular.

- $h'(e)$ is a pair $(M_1, M_2)$ of DPDA's such that $M_e \in \mathcal{E}_{TM} \iff L(M_1) \cap L(M_2) = \emptyset$. Over all inputs $x$.

$G$ generates the set of all "broken computations" of $M_e(x)$. A computation to be valid is like $\ldots x \# x' \# x'' \# x''' \ldots$ where the strings between the hashes are mostly equal to DOUBLE WORD. The complement $\neg \forall w \exists x: \neg \exists y: \neg M_e(y)$ is a CFG. $L(G) \subseteq \Sigma^* \# \Sigma^* \# \Sigma^* \# \ldots$

$h'(e)$ comes from writing $c_{comp}$ as $\overline{x} \# x' R \# x'' R \# x''' R \ldots$

$M_1$ checks $\overline{x}$

$M_2$ checks $x'$

$h$ reduces $\mathcal{E}_{TM}$ to $\text{ALL-CFG}$, so $\text{ALL-CFG}$ is undecidable.

$h'$ reduces $\mathcal{E}_{TM}$ to $\{(G_1, G_2) : L(G_1) \cap L(G_2) = \emptyset\}$

Contrast with BCFG being decidable.

Complexity Theory: No completeness $\Rightarrow$ Thm. (**) And when $L(M_e) \cap L(M_1) \neq \emptyset$, it is uncomputable and like the language of palindromes.

To wit, $L(G) = \{\text{all strings that do not code valid accepting computations of } M_e \text{ on some } x\}$. If $L(M_e) = \emptyset$, then there are all strings parses, so $L(G) = \Sigma^*$. But if $L(M_e) \neq \emptyset$, then $L(G)$ only ships the string code of some accepting computation $\exists x$, it ships it in a way that creates a non-regular language.