### CSE491/596 Categories and Diction, then Examples of Reductions

<table>
<thead>
<tr>
<th>Elements/Objects</th>
<th>Attributes/predicates/verbs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. string = list&lt;char&gt;</td>
<td>(a) &quot;Halts&quot; - 4, a2 &quot;run forever&quot; - 4, not any inst. of 2</td>
</tr>
<tr>
<td>2. Language = set&lt;string&gt;</td>
<td>(b) &quot;Decidable&quot; - 3 and 5</td>
</tr>
<tr>
<td>3. Class = set&lt;Language&gt;</td>
<td>(c) &quot;accepts&quot; - 4</td>
</tr>
<tr>
<td>4. Machine</td>
<td>(d) &quot;be accepted by ...&quot; 1 meaning ( x \in L(M) ), 2 as ( L(M) )</td>
</tr>
<tr>
<td>5. Decision Problem ( \equiv ) Language</td>
<td>meaning &quot;the language [of strings] accepted by a machine&quot;</td>
</tr>
</tbody>
</table>

(e) is c.e. --- machine? class? The person saying "machine" probably meant to allow for the point that a given machine might not halt for all inputs. The person saying "class" either meant that RE is a class of languages, or means that any class of Turing machine languages like P or NP must be a subset of RE. Grammatically, as a matter of diction, only a language can have the attribute of being c.e. A decision problem---?---the preferred term then is partially decidable (on the 'yes' side).

(f) "ends in a '0' " --- ? Strictly it’s only string. But maybe you have in mind the language \( E_0 = \{ x : x \) ends in a 0\}. Or the regular expression \((0 + 1)^*0\).

### Some Common Fallacies:

1. Subsets: "Any subset of a decidable language is decidable." Exposing it: \( \Sigma^* \) is a decidable language, in fact a regular language, but the mega-undecidable language \( ALL_{TM} \) is a subset of \( \Sigma^* \).

2. "If L is undecidable then L is c.e."

3. Intension vs. Extension: "Isn't \( ALL_{TM} \) the same as \( \Sigma^* \)?"

\( ALL_{TM} \) is the language of codes \( \langle M \rangle \) of machines \( M \) such that \( L(M) = \Sigma^* \).

As languages, \( ALL_{TM} \) and \( E_{TM} \) are disjoint, i.e., \( ALL_{TM} \cap E_{TM} = \emptyset \) which is saying that the condition on the set \( \{ \langle M \rangle : L(M) = \Sigma^* \ and \ L(M) = \emptyset \} \) is incompatible.

[The recitation went into a long discussion of the fact of the \( ALL_{TM} \) language not literally "being" \( \Sigma^* \) and why it is a proper subset of \( \Sigma^* \)--because it includes strings like \( \langle M_1 \rangle \) for the machine \( M_1 \) below but not \( \langle M_0 \rangle \) for the machine \( M_0 \) whose language is \( \emptyset \).]

\[
\begin{array}{c}
(0/0, R), \\
(1/1, R)
\end{array}
\quad
\begin{array}{c}
(0/0, R), \\
(1/1, R)
\end{array}
\quad
\begin{array}{c}
M_1 \\
L(M_1) = \Sigma^*
\end{array}
\quad
\begin{array}{c}
M_0 \\
L(M_0) = \emptyset
\end{array}
\]

[The last prepared example of the recitation was about how reductions can be "plus and play" when you vary particulars of what is done before or after a simulation. The idea is to trace out the logical...
analysis that results. It involved the following problem, which was given for homework in a recent year. I originally defined it without the primes, i.e. just saying $M$ everywhere, but explained how that can lead to confusion between the source $M$ in the reduction and the "target property."

**OnlyEps**

INST: A Turing machine $M''$. QUES: Is $L(M'') = \{\varepsilon\}$? That is, does $M''$ accept $\varepsilon$ but no other string?

Here are diagrams of reductions showing $A_{TM} \leq_m \text{OnlyEps}$ and then $D_{TM} \leq_m \text{OnlyEps}$.

- $\langle M, w \rangle \xrightarrow{f} M' = \langle M \rangle \xrightarrow{g} M''$

If $M$ accepts $\varepsilon$, $L(M') = \{\varepsilon\}$ Thus $\langle M, w \rangle \in A_{TM} \implies \langle M' \rangle \in \text{OnlyEps}$

If $M$ does not accept $\langle M \rangle \implies L(M') = \emptyset \implies \langle M' \rangle \notin \text{OnlyEps}$.

$M$ does not accept $\langle M \rangle \implies L(M'') = \Sigma^*$ Thus: $\langle M \rangle \notin D_{TM} \implies \langle M'' \rangle \notin \text{OnlyEps}$

Other variations on the theme can put the test for $x = \varepsilon$ after rather than before:
\( M \) accepts \( w \implies L(M') = \{\epsilon\} \implies M' \in \text{OnlyEps} \\
\langle M, w \rangle \notin A_{\text{TM}} \implies L(M') = \emptyset \implies M' \notin \text{OnlyEps}.

\( M \in K_{\text{TM}} \implies L(M'') = \{\text{all palindromes of length greater the \# of steps \( M \) took to accept \langle M \rangle}\} \implies L(M'') \text{ is nonregular.} \\
M \notin K_{\text{TM}} \implies L(M'') = \emptyset \implies L(M'') \text{ is regular. Thus } K_{\text{TM}} \leq_m I_{\text{REG}}, \text{i.e. } D_{\text{TM}} \leq_m I_{\text{REG}} \\
\text{Thus } I_{\text{REG}} \text{ is not c.e.}

For self-study, do the correctness logic on these reductions. Also make the second one work with the "delay switch" idea. It turns out that the \text{OnlyEps} language is in the least \( \equiv_m \) equivalence class of languages that reduce from both \( K \) and \( D \). In particular, it is lower than \( \text{ALL}_{\text{TM}} \) and \( \text{TOT} \).

[Technically, \text{OnlyEps} and \( K \) and \( D \) are all in the same equivalence class under Alan Turing's original reducibility notion, called \text{Turing reductions} and written \( \leq_T \). But Turing reductions would collapse the left-right dimension (which corresponds to \( \exists \) versus \( \forall \) in logic) down to a single stick, as at right below. So I prefer to avoid them at this point.]

[We can drop the "TM" subscripts not only when the context is clear but because using Java or any other high-level programming language would give exactly the same classification of the analogously-defined languages, e.g. \( A_{\text{Java}}, D_{\text{Java}}, K_{\text{Java}}, \text{OnlyEps}_{\text{Java}}, \) etc. But now we will see machines between Turing machines and DFAs for which the classifications do change and the distinction between "decidable" and "undecidable" is almost on a knife-edge.]
**HW5(1) answer:**

**Reversal**

INST: A Turing machine $M''$.

QUES: Is $L(M'') = (L(M''))^R$?  Note: $\varnothing^R = \{x^R : x \in \varnothing\} = \varnothing$

Here are diagrams of reductions showing $A_{TM} \leq_m OnlyEps$ and then $D_{TM} \leq_m OnlyEps$.

$M$ accepts $w \implies L(M') = \{01\}$ Thus $\langle M, w \rangle \in A_{TM} \implies \langle M'\rangle \notin Reversal$

$\langle M, w \rangle \notin A_{TM} \implies L(M') = \varnothing \implies \langle M'\rangle \in Reversal$.

$M$ accepts $\langle M \rangle \implies L(M'') = \Sigma^*$ Thus: $\langle M \rangle \notin D_{TM} \implies \langle M''\rangle \in Reversal$

$M$ does not accept $\langle M \rangle \implies L(M'') = \{01\}$ Thus: $\langle M \rangle \in D_{TM} \implies \langle M''\rangle \notin Reversal$

Other variations on the theme can put the test for $x = 01$ after rather than before:
\( M \) accepts \( w \Rightarrow L(M') = \{ \epsilon \} \Rightarrow M' \in \text{OnlyEps} \)
\( \langle M, w \rangle \notin A_{TM} \Rightarrow L(M') = \emptyset \Rightarrow M' \notin \text{OnlyEps} \).

\( M \in K_{TM} \Rightarrow L(M'') = \{ \text{all palindromes of length greater the # of steps } M \text{ took to accept } \langle M \rangle \} \Rightarrow L(M'') \text{ is nonregular.} \)
\( M \notin K_{TM} \Rightarrow L(M'') = \emptyset \Rightarrow L(M'') \text{ is regular.} \) Thus \( K_{TM} \leq_m \sim I_{\text{REG}} \), i.e. \( D_{TM} \leq_m I_{\text{REG}} \)
Thus \( I_{\text{REG}} \) is not c.e.

**Tue 11/29/2022 Review Session**

HW5: Alternate way to pad a short clause like \((u \lor w)\):
\((u \lor w \lor z) \land (u \lor w \lor \overline{z}) \cdot (w_0) \text{ becomes } (w_0 \lor z \lor z') \land (w_0 \lor \overline{z} \lor z') \land \ldots \)

\[
\phi = (x_1 \lor \overline{x}_2 \lor x_3) \land (x_1 \lor x_2 \lor \overline{x}_3) \land (\overline{x}_1 \lor \overline{x}_3 \lor \overline{x}_4),
\]
one of them is to set \( x_1 \) true and \( x_3 \) false; then \( x_2 \) and \( x_4 \) become "don't-cares".

In Cook-Levin, the only 3-clauses are ones of the form \((\overline{u} \lor \overline{v} \lor \overline{w})\) and those have the property that they cannot be satisfied \( 3x \), because of the other clauses \((u \lor w \lor z) \) and \((v \lor w \lor z) \).
Edge-Disjoint Paths

The reduction makes $f(\phi) = (G_\phi, s_1, s_2, t_1, t_2)$
Here is the whole thing for the formula used before:

\[ \phi = (x_{11} \lor \bar{x}_{21} \lor x_{31}) \land (x_{12} \lor x_{22} \lor \bar{x}_{32}) \land (\bar{x}_{13} \lor \bar{x}_{33} \lor \bar{x}_{43}) \]

Reduction from \( A_{TM} \), whose instance type is "An \( M \) and a \( w \)".
Example of designing a reduction by putting the correctness logic first (HW3, problem 3):

\[(M, w) \in A_{TM} \iff M \text{ accepts } w \implies M'(x) \text{ visits all of its states (i.e., the states of } M'), \text{ for some } x\]

\[(M, w) \notin A_{TM} \iff M \text{ does not accept } w \implies [\text{for all } x] M'(x) \text{ does not visit all of its states.}\]