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</td>
</tr>
<tr>
<td></td>
<td>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 )</td>
<td>meanigen &quot;the language [of strings] accepted by a machine&quot;</td>
</tr>
</tbody>
</table>

\( \) 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 \( \text{RE} \) is a class of languages, or means that any class of Turing machine languages like \( \text{P} \) or \( \text{NP} \) must be a subset of \( \text{RE} \). 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 \( E_0 = \{ x : x \text{ ends in a } 0 \} \). 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 \( \text{ALL}_{\text{TM}} \) is a subset of \( \Sigma^* \)

2. "If \( L \) is undecidable then \( L \) is c.e."

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

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

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

[The recitation went into a long discussion of the fact of the \( \text{ALL}_{\text{TM}} \) 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 \( \emptyset \).]

\[
\begin{align*}
(0/0, R), \\
(1/1, R) \\
M_1 \\
L(M_1) = \Sigma^*
\end{align*}
\]

\[
\begin{align*}
(0/0, R), \\
(1/1, R) \\
M_0 \\
L(M_0) = \emptyset
\end{align*}
\]

[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 \( L(M'') = \{ \epsilon \} \)? That is, does \( M'' \) accept \( \epsilon \) but no other string?

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

\[
\langle M, w \rangle \xleftarrow{f} M' = \begin{cases} 
\text{Simulate } M(w) \\
\text{if } x \neq \epsilon \text{ reject} \\
\text{if & when it accepts} \\
\text{accept } x.
\end{cases}
\]

\[
\langle M \rangle \xleftarrow{g} M'' = \begin{cases} 
\text{Simulate } M(M) \\
\text{if & when it accepts} \\
\text{accept } x.
\end{cases}
\]

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

\( \langle M, w \rangle \notin A_{TM} \implies L(M') = \emptyset \implies \langle M' \rangle \notin \text{OnlyEps} \).

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

\( M \) does not accept \( \langle M \rangle \implies L(M'') = \{ \epsilon \} \) Thus: \( \langle M \rangle \in D_{TM} \implies \langle M'' \rangle \in \text{OnlyEps} \)

Other variations on the theme can put the test for \( x = \epsilon \) after rather than before:

\[
\langle M, w \rangle \xleftarrow{f} M' = \begin{cases} 
\text{Simulate } M(w) \\
\text{if & when it accepts} \\
\text{if } x = \epsilon \text{ accept} \\
\text{else reject}
\end{cases}
\]

\[
\langle M \rangle \xleftarrow{g} M'' = \begin{cases} 
\text{Simulate } M(M) \\
\text{for up to } n \text{ steps} \\
\text{if it accepts within that time} \\
\text{accept } x \text{ iff } x \text{ is a palindrome} \\
\text{else reject } x
\end{cases}
\]
$M$ accepts $w \implies L(M') = \{e\} \implies M' \in \text{OnlyEps}$

$\langle M,w \rangle \notin A_{TM} \implies L(M') = \emptyset \implies M' \notin \text{OnlyEps}$.

$M \in K_{TM} \implies L(M'') = \{\text{all palindromes of length greater the # of steps } M \text{ took to accept } \langle M \rangle\} \implies L(M'')$ is nonregular.

$M \notin K_{TM} \implies L(M') = \emptyset \implies L(M'')$ is regular. Thus $K_{TM} \leq_m I_{\text{REG}}$, i.e. $D_{TM} \leq_m I_{\text{REG}}$

Thus $I_{\text{REG}}$ 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 $\text{ALL}_{TM}$ 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 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.]