CSE491/596 Categories and Diction, then Examples of Reductions

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

(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:

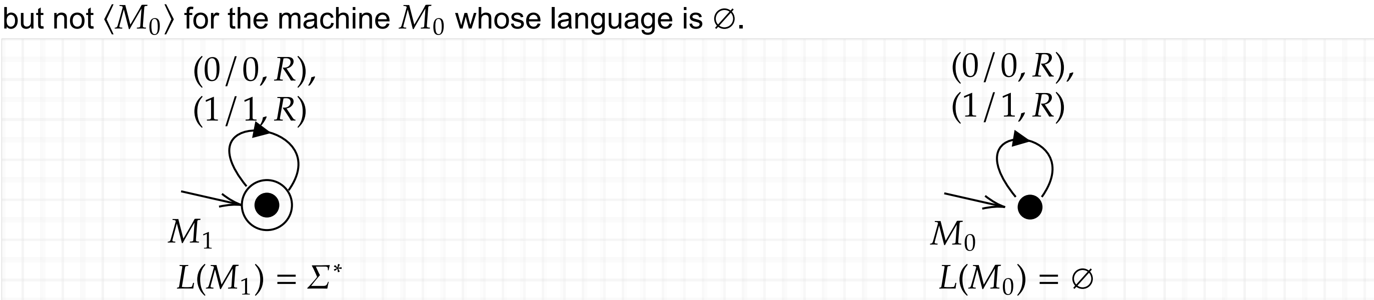
1. Subsets: "~~Any subset of a decidable language is decidable.~~" Exposing it: Σ^* is a decidable language, in fact a regular language, but the mega-undecidable language ALL_{TM} is a subset of Σ^*

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

3. Intension vs. Extension: "Isn't ALL_{TM} the same as Σ^*?"
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 $\left\{\langle M \rangle : L(M) = \Sigma^* \textit{ and } L(M) = \emptyset\right\}$ is incompatible.

[The recitation went into a long discussion of the fact of the ALL_{TM} language not literally "being" Σ^* and why it is a proper subset of Σ^*---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 $\emptyset$.

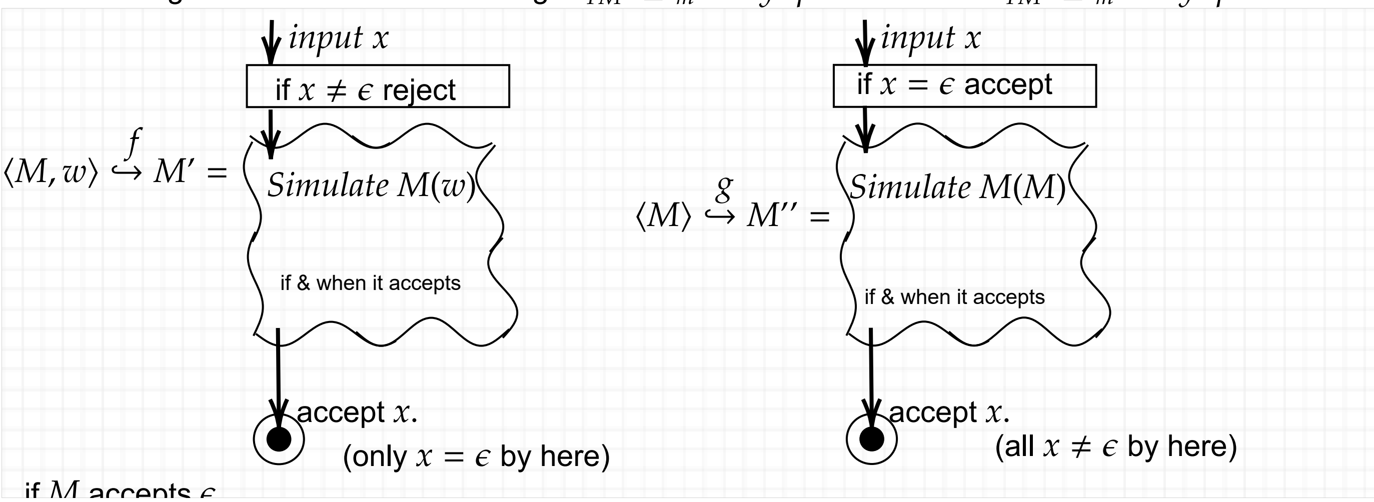


[The last prepared example of the recitation was about how reductions can be "plus and play" when you vary particulars of what is done before or after a simulation. The idea is to trace out the logical analysis that results. It involved the following problem, which was given for homework in a recent 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 ϵ but no other string?

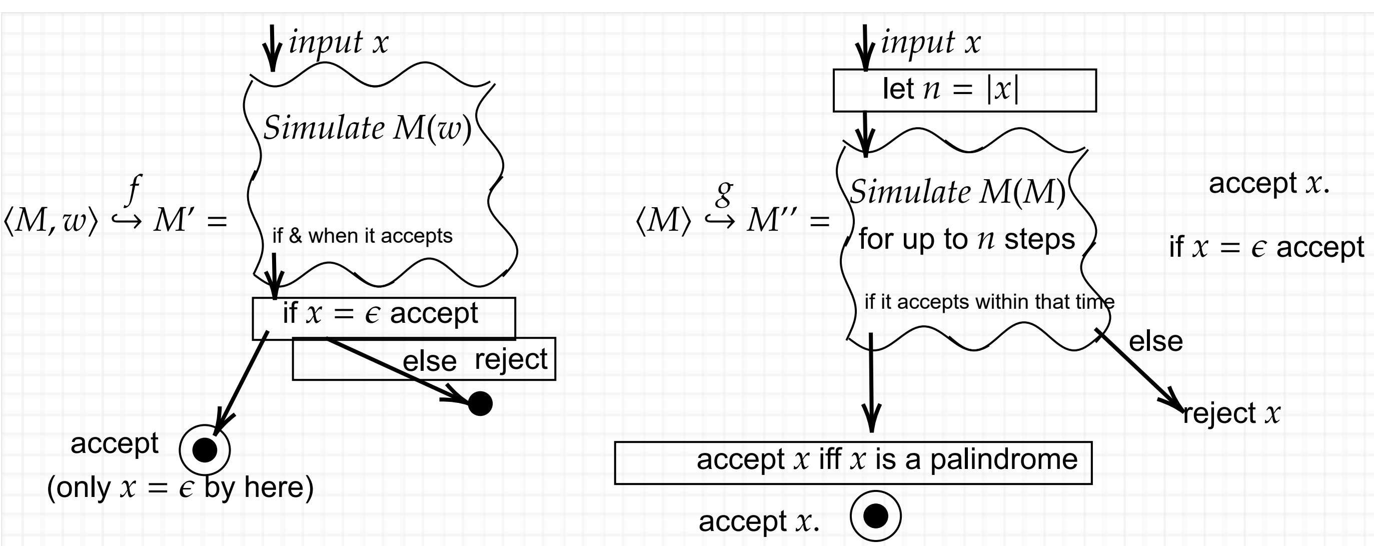
Here are diagrams of reductions showing $A_{TM} \leq_m OnlyEps$ and then $D_{TM} \leq_m OnlyEps$.



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

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

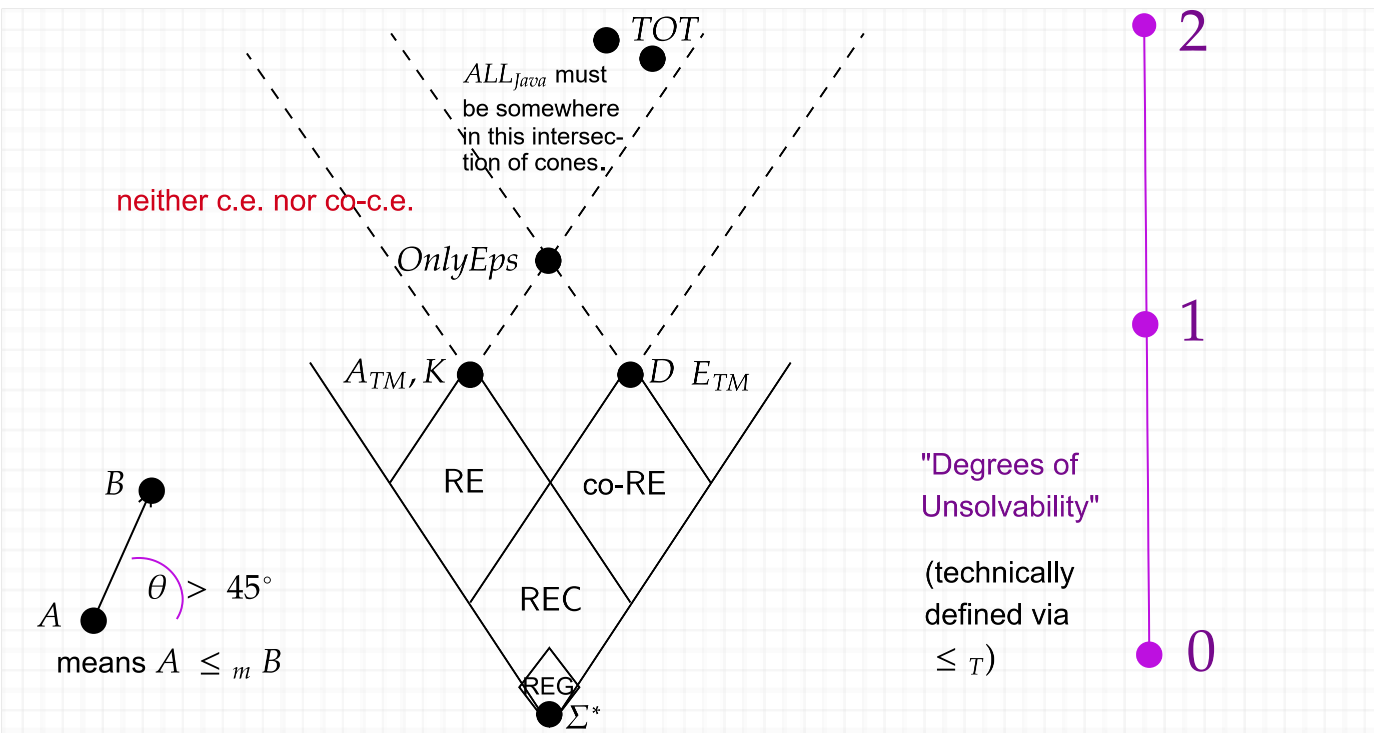
Other variations on the theme can put the test for $x = \epsilon$ after rather than before:



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

$M \in K_{TM} \implies L(M'') = \{$all palindromes of length greater the # of steps M took to accept $\langle M \rangle\}$
$\implies L(M'')$ is nonregular.
$M \notin K_{TM} \implies L(M'') = \varnothing \implies L(M'')$ is regular. Thus $K_{TM} \leq_m \sim I_{REG}$, i.e. $D_{TM} \leq_m I_{REG}$
Thus I_{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 $OnlyEps$ language is in the least $\equiv_m$ equivalence class of languages that reduce from both K and D. In particular, it is lower than ALL_{TM} and TOT. [Technically, $OnlyEps$ 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. $A_{Java}, D_{Java}, K_{Java}, OnlyEps_{Java}$, 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.]