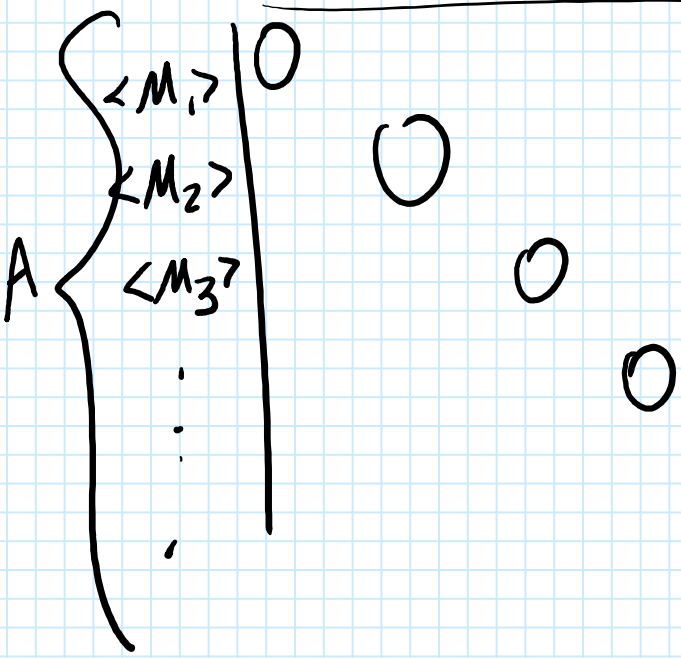
W9L2 decidability, the halting problem, constructions

Wednesday, April 1, 2020 9:19

problem 5 from lab:

$s_1 = \varepsilon$ $s_2 = \ldots$ s_3 s_4 . . . . . } shortlex ordering of Σ^*



Idea: let's build a decider which is not M_1, not M_2, etc.
Here's a new decider B.
B = "On input string w,

1. Run the shortlex enumerator for Σ^* until it prints $s_i = w$.
2. Run the enumerator for A until it prints the i^{th} machine, M_i.
3. Run M_i on input w. Do the opposite."

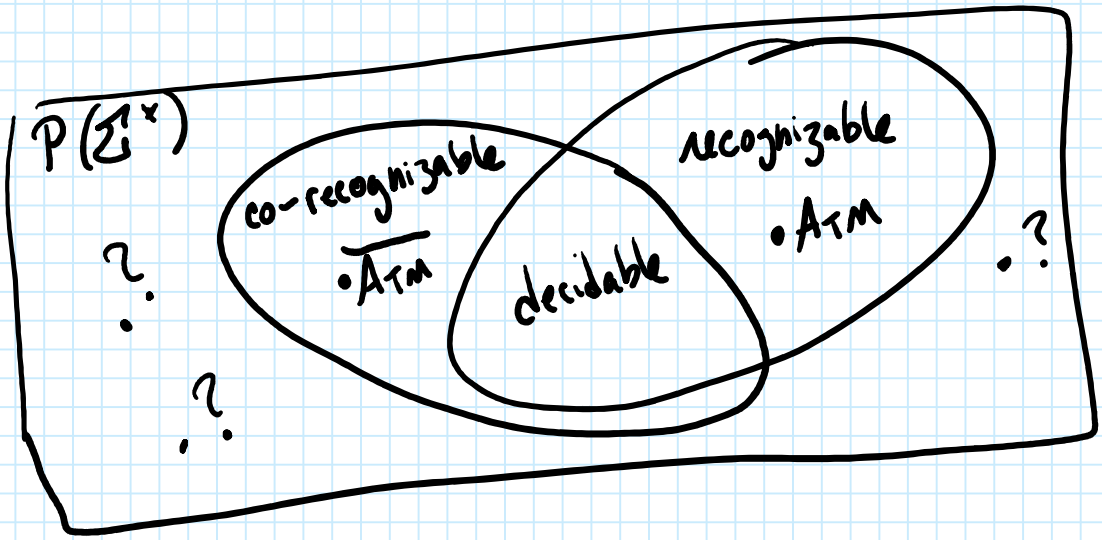
Things to check:
- B is a decider: Line 1 will finish, line 2 will also finish. We know that M_i is a decider, so line 3 also finishes.
- $L(B)$ is not equal to $L(M_1), L(M_2)$, ...: Consider some arbitrary j. If $s_j \in L(M_j)$ then M_j accepts s_j, so B will reject s_j, so the languages are different. If s_j is not in $L(M_j)$ then M_j rejects s_j, so B will accept it, so the languages are different. □

$A_{TM} = \{ < M, w > \mid M \text{ is a Turing machine which accepts string } w\}$
We know: A_{TM} is not decidable, but it is recognizable.

Theorem: A language is decidable if and only if it is both recognizable and co-recognizable.

Def: A language L is co-recognizable if $\overline{L}$ is recognizable.

Claim: $\overline{A}_{TM}$ is not recognizable.
Pf: (by contradiction)
If it were recognizable, then A_{TM} would be both recognizable (from before) and co-recognizable (from this assumption), so then by the theorem, A_{TM} is decidable. But it's not! □



Decidable languages are ones where a TM can say either "yes" or "no", definitively.
Recognizable languages are one where a TM can say "yes" but if the answer is "no" they might loop.
Co-recognizable languages are ones where a TM can say "no" but if the answer is "yes" they might loop.

$$HALT_{TM} = \{ \langle M, w \rangle \mid M \text{ is a Turing machine which halts on inputstring } w \}$$

This is recognizable:

G = "On input $\langle M, w \rangle$ where M is a Turing machine, w is a string:
1. Run M on input w.
2. Accept."

G is a recognizer for $HALT_{TM}$:
If $< M, w > \in HALT_{TM}$ then M halts on w, so G will finish line 1 and get to line 2, and accept. Thus $HALT_{TM} \subseteq L(G)$.
If G accepts $< M, w >$ then G reached line 2, so line 1 must have finished. Thus M halted on w. So $< M, w > \in HALT_{TM}$, so $L(G) \subseteq HALT_{TM}$. □

Claim: $HALT_{TM}$ is not decidable.

Proof: (by contradiction)
Assume that $HALT_{TM}$ is decidable, and is decided by some TM called R.
We will show how, using R, to build a decider S for A_{TM} (this will be our contradiction!).

Build S = "On input $\langle M, w \rangle$ where M is a Turing machine and w is a string:
1. Run R on input $\langle M, w \rangle$. If R rejects, we also reject.
2. If R accepts, then run M on input w.
3. If M accepted, accept. Else, if M rejected, reject."

S is a decider: line 1 finishes because R is a decider. Line 2 finishes, if we run it, because the only way we run it is if we already know (from R) that M will halt on w. Line 3 also halts.

If $<M, w> \in A_{TM}$ then M accepts w so M halts on w, so on line 1, R accepts; on line 2, M accept, on line 3, S accepts overall.
If $<M, w>$ not in A_{TM}, then either M loops on w (in which case, S will reject on line 1) or M rejects w (in which case, S will reject on line 3).

$\Rightarrow\Leftarrow$ □