COMPUTATION HW4

Name	ID
Gur Telem	206631848

1. Question

1.1. $L_1 = \{(\langle M_1 \rangle, \langle M_2 \rangle) \mid \exists w : f_{M_1}(w) = f_{M_2}(w)\}$. RTP: $L_1 \in RE \backslash R$.

Proof. We'll define a TM M that will accept the language L_1.

Let $(\langle M_1 \rangle, \langle M_2 \rangle)$ a pair of encoded TMs.

Using a controlled run, we can run simultaneously on both M_1 and M_2 on each $w_i \in \Sigma^*$ and if both M_1, M_2 halt and output the same value, then M will accept w_i. Let's describe it formally.

M run is:

For each $i \in \{1, 2, 3, \dots\}$, take $w_i \in \Sigma^*$ (assume w_i is the i-th word in the lexicographical order of Σ^*)

Run each of M_1 and M_2 on each of $w_1, w_2, \dots, w_i$ for i steps.

If for a word w_j both M_1 and M_2 halted and returned the same value, accept $(\langle M_1 \rangle, \langle M_2 \rangle)$ (go to q_{acc}).

If not, continue to the next i.

If $\exists w_i \in \Sigma^*$ s.t. both machines will halt (mark the number of steps until they both halt k) and return the same value, then after $\max\{i, k\}$ iterations of M's algorithm, we will find it and accept the pair $(\langle M_1 \rangle, \langle M_2 \rangle)$.

If no such w_i exists, then M will continue to run forever for the pair, meaning it will not accept it.

Thus we proved that $L(M) = L_1$.

Thus $L_1 \in RE$ (there's a TM which accepts it).

Now we want to prove that $L_1 \notin R$.

For that we can use the classic move of creating a reduction from $HP \leq L_1$.

The reduction is defined as follows: $f(\langle M \rangle, x) = (\langle M_x \rangle, \langle M_x \rangle)$ s.t. M_x is the TM that **ALWAYS** runs M on x (whatever the input given to it).

First, let's prove it's a valid reduction. The function is full (as we know, if $\langle M \rangle$ isn't a valid encoding, we see it as M_{stam} which runs forever). Also, we know that the function is computable as we saw in the lecture.

And now, assume $(\langle M \rangle, x) \in HP$. Then it means that M halts for x (and returns whatever value). Thus $\langle M_x \rangle$ will always halt and always return the same value. So we got that $f_{M_x}(x) = f_{M_x}(x)$ meaning that $f(\langle M \rangle, x) = (\langle M_x \rangle, \langle M_x \rangle) \in L_1$.

And the other direction: Let $(\langle M \rangle, x)$ be a TM and an input. Assume $f(\langle M \rangle, x) = (\langle M_x \rangle, \langle M_x \rangle) \in L_1$. Thus M_x is a TM that runs M on x and halts (with whatever value). And thus $(\langle M \rangle, x) \in HP$.

Thus this is a valid reduction. And from the theorem we learned in class, since $HP \notin R \Longrightarrow L_1 \notin R$. Thus (along with $L_1 \in RE$) we got that $L_1 \in RE \backslash R$ as we wanted.

Q.E.D □

1.2. $L_2 = \{(\langle M_1 \rangle, \langle M_2 \rangle) \mid \forall w : f_{M_1}(w) = f_{M_2}(w)\}$. RTP: $L_2 \notin RE$.

Proof. We'll use $L_{\Sigma^*} \notin RE$ that we saw in the lecture.

And ofc we're going to define a reduction from $L_{\Sigma^*} \leq L_2$ defined with $f(\langle M \rangle) = (\langle M' \rangle, \langle M' \rangle)$ with M' defined as such:

Run M on the input.

If the input was rejected, enter an infinite loop.

If the input was accepted, M' will output 1.

So f is computable. It's also full because whatever M is, we can run it (remember universal machines). Meaning the function is computable and full. Thus we only need to show $f(\langle M \rangle) \in L_2 \Longleftrightarrow \langle M \rangle \in L_{\Sigma^*}$.

Let M a TM.

Assume $\langle M \rangle \in L_{\Sigma^*}$.

Let $w \in \Sigma^*$.

Thus M' as we defined it, will halt in step 1 for w and will accept. Thus M' will output 1 in step 3. Thus $f_{M'}(w) = 1 = f_{M'}(w) \Longrightarrow f(\langle M \rangle) = (\langle M' \rangle, \langle M' \rangle) \in L_2$.

Now assume $f(\langle M \rangle) = (\langle M' \rangle, \langle M' \rangle) \in L_2$.

Let $w \in \Sigma^*$. $f_{M'}(w) = f_{M'}(w)$ means that M' halts for w. From out definition of M', this can only happen in step 3. Thus M halted and accepted w. And since that's true for every w, then M accepts every w which means $L(M) = \Sigma^* \Longrightarrow \langle M \rangle \in L_{\Sigma^*}$.

Thus $f(\langle M \rangle) \in L_2 \Longleftrightarrow \langle M \rangle \in L_{\Sigma^*}$. And this means that $L_{\Sigma^*} \leq L_2$ is a valid reduction.

Since we know that $L_{\Sigma^*} \notin RE$ (see the ezer.pdf on the course's website) it means (from the reduction's theorem) that $L_2 \notin RE$.
Q.E.D

□

1.3. $L_3 = \{\langle M \rangle \mid \forall M', w : f_M(\langle M' \rangle, w) \in \{0,1\} \wedge (M' \textbf{ stop for w} \iff f_M(\langle M' \rangle, w) = 1)\}$**.** RTP: $L_3 = \emptyset \in R$ because M_{rej} which rejects everything immediately decides $\emptyset$.

Proof. Let's prove though that $L_3 = \emptyset$ (we actually proved most of it in the lecture, I realise in retrospect so it's probably similar to that proof so you can skip to the last line if you want).
BWoC $L_3 \neq \emptyset$.
Let $M \in L_3$.
We saw in the lecture that $RE \backslash R$ isn't an empty set. So let's take any TM from it:
Let $M_{RE} \in RE \backslash R$.

Claim 1. Thus $L(M_{RE})$ isn't decidable.

Let's define a new TM. for each w input, M_d is defined as such:
Checks if $f_M(\langle M_{RE} \rangle, w) = 1$ and if not, it halts and rejects w.
Runs w on M_{RE} and returns the same answer as M_{RE}.
From definition M halts on the input $(\langle M_{RE} \rangle, w)$ (because $f_M(\langle M_{RE} \rangle, w)$ is well defined for each TM and input).
Case 1: M_{RE} doesn't halt for w. Then $f_M(\langle M_{RE} \rangle, w) = 0$ and M_d will halt (reject) in step 1.
Case 2: M_{RE} halts for w. Then $f_M(\langle M_{RE} \rangle, w) = 1$ and M_d will halt in step 2 while running M_{RE} on w.
So we got that M_d halts for every input.
Additionally, $w \in L(M_{RE}) \iff M_{RE} \text{ accepts } w \implies f_M(\langle M_{RE} \rangle, w) = 1 \implies f_{M_d}(w) = f_{M_{RE}}(w) \implies w \in L(M_D)$.
But from the definition of M_d: $w \in L(M_d) \implies w \in L(M_{RE})$. So we got $L(M_{RE}) = L(M_d)$.
So since M_d accepts $L(M_{RE})$ and M_d halts for every input, we got that $L(M_{RE})$ is decidable in direct contradiction to our assumption.
Thus $L_3 = \emptyset$.
And we know that $L_3 = \emptyset = L(M_{rej}) \in R$ (every finite language is in R).
Q.E.D □

2. Variations on Rice's Theorem

2.1. For coRE.

2.1.1. *For S trivial attribute.* For $S = \emptyset$ RTP: $L_S \in R$.

Proof. Meaning that $L_S = \emptyset$ (if $\langle M \rangle \in L_S$ then $L(M)$ whatever it is will be in S in contradiction to it being an empty set).

And we know that $\emptyset \in R$ (for example because it's a finite set). □

For $S = coRE$ RTP: $L_S \notin R$.

Proof. Let M be TM.

Assume $\langle M \rangle \in L_S$ so $L(M) \in S$.

But if $L(M) \in S \Longrightarrow L(M) \in RE$ (because the language has M which accepts it).

Thus, if we mark $S' = R \subseteq coRE$ and also $S' \subseteq RE$.

And because $L(M) \in S \wedge L(M) \in RE \Longrightarrow L(M) \in R \Longrightarrow \langle M \rangle \in L_{S'}$.

So we got that $L_S \subseteq L_{S'}$. Since there's a TM M' which accepts $HP \in RE$ but $HP \notin R$. So $HP = L(M') \notin S' = R$. But also $\langle M \rangle \in L_{RE}$ per definition.

Thus we got that $L_{S'} \subset L_{RE}$. We from rice, we know that not only $L_{S'} \notin R$ but also $L_{S'} \notin RE$ (because $\emptyset \in S' = R$).

Also, since $R \subset coRE$, if M'' is a TM such that $\langle M'' \rangle \in L_R \Longrightarrow L(M'') \in R = S' \subset S = coRE \Longrightarrow \langle M'' \rangle \in L_S$. Thus $L_{S'} \subseteq L_S$ and thus $L_{S'} = L_S$.

Thus we got that $L_S \notin RE$ (let alone in R).

We proved that $L_S \notin R$. □

2.1.2. *Write and prove a theorem for attributes $S \subseteq coRE$ s.t. $L_S \notin R$.*

Theorem 2. *Let $S \subseteq coRE$ s.t. $S \cap R \neq \emptyset \Longrightarrow L_S \notin R$.*

Proof. Assume $S \subseteq coRE$.

In the previous section we proved that $L(M) \in coRE \Longrightarrow L(M) \in R$, this ofc applies for every $L(M) \in S$ as a subset of $coRE$.

Let M s.t. $\langle M \rangle \in L_S$ so $L(M) \in R$.

And similarly to how we proved in the first section $L_S \subseteq L_R$. Thus $L_{S \setminus R} = L_S \setminus L_R = \emptyset$ (directly from expanding the definition of $L_?$).

So $L_S = L_S \cap L_R = \{\langle M \rangle \mid L(M) \in S \wedge L(M) \in R\} = \{\langle M \rangle \mid L(M) \in S \cap R\} = L_{S \cap R}$.

And since $S \cap R \subset RE$ (and because $S \cap R \neq \emptyset$) then $S \cap R$ is a non trivial attribute as defined in the original Rice's theorem. Thus $L_{S \cap R} \notin R$. And since $L_S = L_{S \cap R} \Longrightarrow L_S \notin R$.

Q.E.D

Note: It's also true that if $\emptyset \in S$ then $L_S \notin RE$ and is easy to prove. □

2.2. For functions.

2.2.1. *RTP: For trivial attribute S, $L_S \in R$.*

Proof. Case 1: $S = \emptyset$.

So $L_S = \emptyset$ because for each M, $f_M \notin S$ so $\langle M \rangle \notin L_S$. And we know that $L_S \in R$.

Case 2: $S = \mathcal{F}$.

Let M be a TM, then $f_M \in \mathcal{F} = S$. Thus $\langle M \rangle \in L_S \Longrightarrow L_S = \{\langle M \rangle \mid \forall M\}$. Thus M_{acc} will decide L_S as it will accept every TM (and will ofc halt for each).

Q.E.D □

2.2.2. *RTP: Rice's theorem for functions. Can we add condition for S s.t. $L_S \notin RE$?*

Proof. Let $S \subseteq \mathcal{F}$. Assume $S \notin \{\emptyset, \mathcal{F}\}$.
We need to prove that $L_S = \{\langle M \rangle \mid f_M \in S\} \notin R$.
Let $f \in L_S$ (there's one because $S \neq \emptyset$). Note that f is computable. So there's a TM which calculates it.
We'll split into cases.

Case 1. $f_{M_{stam}} \notin S$
We'll define a reduction $HP \leq L_S$ with function f_r.
So for $f_r(\langle M \rangle, x) = \langle M_x \rangle$ with M_x defined for input w as such:
(1) Run M on x.
(2) Write to the tape $f(w)$ and halt.
Since f in computable, f_r is a full function and computable.
So we need to prove $(\langle M \rangle, x) \in HP \iff f(\langle M \rangle, x) \in L_S$.
Let $(\langle M \rangle, x)$.
$\implies$: Assume $(\langle M \rangle, x) \in HP$. Thus, M_x will halt sometime during step 1 and continue to step 2. On step 2 it will return $f(w)$.
Thus $f_{M_x} = f \in L_S$ (under our assumption). So we got $(\langle M \rangle, x) \in HP \implies f((\langle M \rangle, x)) \in L_S$.
$\impliedby$: Assume $f((\langle M \rangle, x)) = \langle M_x \rangle \in L_S$.
Since $f_{M_{stam}} \notin S \implies M$ halts for x (otherwise f would be a function that is defined for no value as it would never halt).
Thus $(\langle M \rangle, x) \in HP$.
Thus the reduction is a valid one and since $HP \notin R \implies L_S \notin R$ from the reduction's theorem we learned in class.

Case 2. $f_{M_{stam}} \in S$
Since $S \neq \mathcal{F}$ then there's a computable function $f' \notin S$.
We'll show a reduction from $\overline{HP} \leq L_S$ function f_r. $f_r((\langle M \rangle, x)) = \langle M_x \rangle$ s.t. M_x is defined for w as such:
(1) Run M on x.
(2) Write to the tape $f'(w)$ and halt.
Since f' is computable then f_r is full and computable.
$\implies$: Assume $(\langle M \rangle, x) \in \overline{HP}$.
Then M_x will never halt in step 1.
So f_{M_x} is a function that is not defined for any input.
Meaning $f_{M_x} = f_{M_{stam}}$. And as such $\langle M_x \rangle \in L_S$.
$\impliedby$: Assume $f((\langle M \rangle, x)) \in L_S$.
BWoC, M halts on x as input.
Thus M_x will always halt on step 1 and thus always return the same value as f'.
So $f_{M_x} = f' \notin S$. Thus $\langle M_x \rangle \notin L_S$ contradiction.
So we got that M doesn't halt for x meaning that $(\langle M \rangle, x) \in HP$.
This proves that the reduction is a valid one. And since $\overline{HP} \notin RE \implies L_S \notin RE \implies L_S \notin R$.

This concludes that proof that $L_S \notin R$.
We also found the condition for which $L_S \notin RE$ ($f_{M_{stam}} \in S$).
Q.E.D □

2.3. **Let $S \subseteq RE$ non trivial s.t. $\Sigma^* \notin S$. RTP: $L_S \notin RE$.**

Proof. Since S is non trivial, let M' s.t. $\langle M' \rangle \in L_S$. From the assumption, $L(M') \neq \Sigma^*$.
We'll prove it using a reduction $\overline{HP} \leq L_S$ with the function $f((\langle M \rangle, x)) = \langle M_x \rangle$ with M_x for input w is defined as such:

(1) Run M' on w. If it accepted, halt and accept w.
(2) Run M on x.
(3) Accept w.

As we see, the function is computable and full. Need to prove $f((\langle M \rangle, x)) \in L_S = \langle M_x \rangle \in \overline{HP}$.
$\implies$ Assume $f((\langle M \rangle, x)) \in L_S$.
So $\Sigma^* \neq L(M_x) \in S$. Thus let $w \in \Sigma^*$ be $w \notin L(M_x)$. BWoC: M halts for x.

So when running M_x, during step 1 we won't accept and go to step 2.
In step 2, we will finish eventually step 2 and go to accept in step 3.
We got $w \in L(M_x)$. In contradiction. Thus M doesn't halt for x.
$(\langle M \rangle, x) \in \overline{HP}$.
$\Longleftarrow$ Assume $(\langle M \rangle, x) \in \overline{HP}$.
Since $L(M') \neq \Sigma^*$ let $w \notin L(M')$.
Thus M_x will not halt in step 1.
And in step 2 it will get stuck in a loop (infinite). Thus $w \notin L(M_x)$.
Now let $w' \in L(M')$. Then M_x will halt in step 1 and accept w'.
So $w' \in L(M_x)$.
We got that $w \in L(M_x) \Longleftrightarrow w \in L(M')$.
Thus $L(M_x) = L(M') \in S \Longrightarrow \langle M \rangle \in L_S$.
This concludes the reduction. So from the reduction's theorem, we get that since $\overline{HP} \notin RE \Longrightarrow L_S \notin RE$.
Q.E.D $\square$

3. Limits on memory and computable functions

For each function, decide if they're computable.

3.1. f_1 on $\langle M \rangle$ will return the index of the right most cell that M will reach during its execution on ε.

Yes! It's computable.

Since we don't need to prove I will explain in short why it's computable.

We can build a TM with a few tapes:

(1) For executing M
(2) For saving all the configurations that we saw M in
(3) Tape to save the max index M reached during execution

Our TM will run M step by step, and for each step:

(1) If M halted, return the max index.
(2) Update the max index if needed.
(3) Check if the current configuration is one that we already saw, if not, save the configuration to the configurations' tape.
(4) If we **did** see the current configuration before, return the current max index.
(5) Continue to the next step of M.

If M has a max index, then there are only a constant number of configurations it can be in. Thus, eventually M will either halt or get into a configuration it was in before. Either way we'll halt out TM and return the max index.

3.2. $f_x(n) = \max\{f_1(\langle M \rangle) : M \in E_n\}$ with E_n a set of all the TM with n states that stop on ε.

Yes! It's computable.

The function is computable. Since E_n is a finite set for each n.

For each $M \in E_n$ it has n states. There's a finite number of ways to define it (for each of the 3 reading options, there are 3 options for the writing for each, there are 3 options for movement of the reading head, for each, there's $n-1$ options for the next state).

Thus E_n is a finite set of TMs. So we can run the function from section 1 on each simultaneously (one step on each $M \in E_n$ at a time).

Since all of the TMs halt on ε, then f_1 will eventually halt with a return value. Then we can compare all the values that we got and return the max.

4. Flexible acceptance

4.1. Need to PROVE: $RE \subseteq A$.

Proof. Let $L(M) \in RE$.

For each state q in M, we're add one more state q' which will mean "go immediately left".

Whenever M is supposed to do an S step and go to state q while writing t to the tape: replace that with writing t to the tape, doing an R step, and moving to state q' instead of q.

If M is in state q', it will do an L step, not changing the tape and moving to state q.

We got an M' which is equivalent to M in the sense that instead of all the S steps in M, M' will go right and then immediately left, then continue as M would.

Thus $L(M') = L(M)$. But since M' doesn't do any S steps (let alone an infinite number of them), if $w \in Flex - L(M')$ then $w \in L(M')$. And since $w \in L(M')$ per definition means $w \in Flex - L(M')$.

Then we got that $Flex - L(M') = L(M')$.

So $L(M') \in A$.

Q.E.D □

4.2. Need to prove $\overline{HP} \in A$.

Proof. Let $(\langle M \rangle, x)$.

We'll define a TM M' on $(\langle M \rangle, x)$ as such:

1. For $i \in \{1, 2, \dots\}$
   - (a) Run M on x for i steps. If M halted, reject $(\langle M \rangle, x)$
   - (b) Do an S step

Assume $(\langle M \rangle, x) \in \overline{HP}$.

Thus during the execution of M', M will never reject in step 1.a. So for each i we'll do an S step and continue to the $i+1$th iteration.

i.e. since we never halt but at the same time, we never get stuck on a single step, we will do an infinite number of S steps.

Thus $w \in Flex - L(M')$.

Assume $(\langle M \rangle, x) \notin \overline{HP}$.

Thus M will halt on x. Let's mark k the number of steps M will do before halting.

As we said, no iteration gets stuck because each sub-step has a finite limit on its execution steps.

Thus in the kth iteration, M will halt on x and reject $(\langle M \rangle, x)$. Meaning that $(\langle M \rangle, x) \notin Flex - L(M')$ because there are a finite number of (S) steps and neither did M' accept $(\langle M \rangle, x)$.

Together, we got that $(\langle M \rangle, x) \in \overline{HP} \iff (\langle M \rangle, x) \in Flex - L(M')$.

Then we showed M' TM s.t. $Flex - L(M') = \overline{HP}$. Thus $\overline{HP} \in A$.

Q.E.D □

4.3. RTP: $L_{=3} \in A$.

Proof. Let's build M for which $Flex - L(M) = L_{=3}$.

Let M' be a TM.

As we proved in the first section, for each TM, there's an equivalent TM without S steps. So let's just assume M' is without S steps (if it includes them, then our M will do the "right then left" trick instead of doing an S step).

So our M running on $\langle M' \rangle$ will be defined as such:

1. For $i \in \{1, 2, \dots\}$, w_i is the i-th word in lexicographic order in Σ^*.
   - (a) Set counter $n = 0$
   - (b) Run M' on each of $w_1, w_2, \dots, w_i$ one by one for i steps (each). For each w_j, if M' accepted it, increase n by 1.
   - (c) If $n == 3$, do an S step.
   - (d) If $n > 3$, reject $\langle M' \rangle$.

Note that we only halt M in step 1.d (in which we reject $\langle M' \rangle$).

Case 1. $|L(M')| = 3$. Assume we're in the ith iteration, then at most 3 of the word $w_1, \ldots, w_i$ can be accepted (M' doesn't accept more than that...).

Thus n will never be set to $n > 3$ and we will never reject (and will never halt).

Now, each of the 3 words in $L(M')$ has a number of steps before M' halts. Let us mark k the max of those numbers.

Again since only 3 words, let us mark t the word with the max lexicographic index.

So from the $\max(k, t)$-th iteration and all iterations following, we will find all of the 3 words with each iteration. Thus making an S step.

Since we never halt, we will do an infinite number of iterations (each step in each iteration has a finite number of execution steps), we will do an infinite number of S steps.

So we got that $\langle M' \rangle \in Flex - L(M)$.

Case 2. $|L(M')| < 3$. In this case we will never reach $n == 3$ do we will never halt (let alone accept), nor we will do any S steps. Thus $\langle M' \rangle \notin Flex - L(M)$.

Case 3. $|L(M')| > 3$. Thus eventually M will find the 4-th word and halt, rejecting $\langle M' \rangle$. So again $\langle M' \rangle \notin Flex - L(M)$.

Thus we got that $|L(M')| = 3 \iff \langle M' \rangle \in Flex - L(M)$.

Q.E.D $\square$

5. Language Classification

For each of those, decide if they're in R or $RE \backslash R$ or not in RE at all.

5.1. $L_1 = \{\langle M \rangle \mid M$ **is a NDTM and there's a calculation path of** M **on** ε **which doesn't cross cell** $10\}$**.** RTP: $L_1 \in R$.

Proof. Let M a NDTM.

We'll define M' NDTM on $\langle M \rangle$ as such:

(1) Write ε to the input of M
(2) Set an area for past configurations and set an area to save current cell index of M
(3) From the current configuration, run a single step of M (here M' branches together with M)
(4) If M passed the 10th cell, we **halt and reject**.
(5) If M halted, then **halt and accept**.
(6) Check if the current configuration of M is one that we already visited. If it is, **halt and accept**.
(7) Save the current configuration of M in the area we set for configurations.
(8) Go to step 3.

If M' will accept $\langle M \rangle$, then it's only in either step 5 or 6.

Since M' branching follows that of M, then it means we found a path with M not passing cell 10. The second we pass cell 10, we halt that path and we don't continue the execution of M. Thus if we reached one of the above steps, it means that in the path of execution of M we didn't pass cell 10. And we either got stuck in an infinite loop (a configuration we already visit in that path of execution) or M halted on ε.

Either way, it's a pass that will never pass cell 10. Thus if M' accepts $\langle M \rangle$ then there must be a path where M doesn't pass cell 10.

And for the other direction:

Assume M' doesn't accept $\langle M \rangle$. All paths of execution of M' either rejected in step 4 (meaning that we that path of M passes cell 10) or never halt.

BWoC M' has a path that doesn't halt:

Thus there's a path of execution for M that doesn't pass cell 10, doesn't halt **and** doesn't repeat a configuration. This we already know is impossible (finite number of configurations that don't include cells after cell 10).

Thus if M' doesn't accept $\langle M \rangle$ is because all paths of executions pass cell 10.

Thus $\langle M \rangle$ is accepted by M' if and only if M has a path of execution that doesn't pass cell 10.

We learned in class that there's a M'' equivalent to M' but deterministic.

Thus M'' is a TM that halts for every input and accepts L_1.

Thus $L_1 \in R$.

Q.E.D □

5.2. $L_2 = \{\langle M \rangle \mid M$ **a NDTM and there isn't a finite path of execution for** M **on** $\varepsilon\}$**.** RTP: $L_2 \notin RE$.

We'll use a reduction $L_{\leq 3} \leq L_2$ defined with $f(\langle M \rangle) = \langle M' \rangle$ a NDTM (with all branchings to identical configurations). M' is defined as such (regardless of the input):

(1) For $i \in \{1, 2, \dots\}$, w_i is the i-th word in lexicographic order in Σ^*.
   (a) Set the counter $n = 0$.
   (b) Run M on each of $w_1, w_2, \dots, w_i$ one by one for i steps (each). For each w_j, if M accepted it, increase n by 1.
   (c) If $n > 3$ then halt and reject.

Proof. Since we want an NDTM, we define it as such that all the branches run the same configuration on the same index of step.

This is a computable and full function.

Note that every step in each iteration is bound to a finite execution time. Thus we will either halt or for each i, we will eventually reach the ith iteration.

$\implies \langle M \rangle \in L_{\leq 3}$.

Thus, in no iteration we will find $n > 3$ as the words $w_1, w_2, \dots, w_i$ are different from each other. Thus step c will never halt the execution.

So since we don't halt the operation of M' at any other point, M' will never halt (all branches are identical, so there won't be even one finite path).

Making $\langle M' \rangle \in L_2$.

$\Longleftarrow f(\langle M \rangle) = \langle M' \rangle \in L_2$.

Meaning that all paths of execution are infinite.

Since all paths are identical as we defined them, it means that we never halt in step c.

BWoC, there are at least 4 words in $L(M)$.

Mark with k the max between the number of iterations required for each of the first 4 words, and the lexicographical index of the last of those 4 words.

So in the kth iteration, we will get that M accept 4 of the w_i(s). Thus making $n > 3$ meaning that M' will halt.

Thus $L(M)$ contains at most 3 words. Meaning that $\langle M \rangle \in L_{\leq 3}$.

Thus we concluded the proof that it's a valid induction and from the induction's theorem, and since $L_{\leq 3} \notin RE$, we get that $L_2 \notin RE$ either.

Q.E.D □

5.3. $L_3 = \{\langle M \rangle \mid M$ **a NDTM and there is a path of execution for** M **on** ε **which repeats a config**$\}$**.** Almost identical to the proof from section 5.1. Except that we reject it we halted copy pasting and changing what is needed:

Proof. Let M a NDTM.

We'll define M' NDTM on $\langle M \rangle$ as such:

(1) Write ε to the input of M
(2) Set an area for past configurations
(3) From the current configuration, run a single step of M (here M' branches together with M)
(4) Check if the current configuration of M is one that we already visited. If it is, **halt and accept**.
(5) If M halted, then **halt and reject**.
(6) Save the current configuration of M in the area we set for configurations.
(7) Go to step 3.

If M' accepts $\langle M \rangle$, then it's only in step 5.

Since M' branching follows that of M, then it means that we repeated a configuration.

And for the other direction:

Assume M has a repeating configuration.

Thus, assume the repeating of the configuration is in the kth branching of M's execution.

In that case, since each step is capped to a finite execution time, we will keep branching and eventually reach the kth branch of execution of M.

At that point, one of the branches of execution of M' will see the repeating configuration of M and will halt, accepting $\langle M \rangle$. Thus there's a path of execution where M' accepts $\langle M \rangle$.

In conclusion M has a repeating configuration $\Longleftrightarrow M'$ will accept $\langle M \rangle$.

Thus $L(M') = L_3$.

Thus $L_3 \in RE$.

Let's prove that $L_3 \notin R$ using a reduction $HP \leq L_3$. $f((\langle M \rangle, x)) = \langle M_x \rangle$ with M_x NDTM defined as such (no matter what the input of it):

We already shows how we detect repeating configuration so we'll just assume we have that ability:

(1) Run M on x for one step.
(2) If it reached a configuration that we saw before, keep going right forever.
(3) If M halted, make an S step staying in the same state and don't change the tape, basically do the step again (stay in the same configuration of M_x itself).
(4) Go to step 1 to continue the execution of M on x.

Since we want M_x to be NDTM we can just set every branching as branching into identical configurations.

Now this is a full and computable function. Assume $f(\langle M\rangle, x) = \langle M_x\rangle \in L_3$. Meaning that M_x repeated a configuration.

If M repeats a configuration, then M_x will keep going right forever (never repeating a configuration).

So in step 1 and 2 it's impossible for M_x to repeat a configuration.

BWoC we reached step 4 with the same configuration twice: Thus M didn't halt and didn't repeat a configuration (otherwise we wouldn't reach step 4). Thus the configuration of M' it new since we it need to hold a config of M that is new. Thus in step 4 we only see new configurations.

Thus if we repeated a configuration, it's in step 3 and only if M halted. Thus M halts on x and thus $(\langle M\rangle, x) \in HP$.

Now assume $(\langle M\rangle, x) \in HP$.

Since M halts on x it never repeats a config. Thus, until M halts, $M_x$will never repeat a config. Thus then M halts, M_x will enter an infinite loop staying in the same configuration (repeating a configuration).

Thus we proved that $(\langle M\rangle, x) \in HP \iff \langle M_x\rangle \in L_3$.

But since $HP \notin R$ (from the reduction's theorem) we get that $L_3 \notin R$ as well.

Q.E.D $\square$

6. Kolmogorov Complexity on a finite set

Given L language:

$$f_L(x) = \begin{cases} K(x) & x \in L \\ 0 & else \end{cases}$$

With $K(x)$ Kolmogorov Complexity.
RTP: $\forall L$ finite language, f_L in computable.

Proof. Let L be a finite language and mark n the number of words in it.
For each of those $w_i \in L$ there's $K(w_i) = t_i$.
Since there is only a finite number of values w_i, t_i we can pair them together and encode them in the states of a TM M s.t. M on input x is defined as follows:

(1) For $i \in \{1, \ldots, n\}$ (note it's a finite number of iterations)
  (a) Take the ith word in L (lexicographic order): w_i.
  (b) Compare x and w_i. If they differ, skip to the next iteration.
  (c) (if not) halt and return the value that is paired to w_i: t_i.
(2) If we didn't halt until now, halt and return 0.

It's obvious that this is a valid TM.
Let $w_i \in L$. Thus in one of the n iterations of M, we will compare w_i to itself (as w_i is encoded in M as it's in L) and then we return the $K(w_i) = t_i$ that was paired to w_i. Thus, for $f_M(w_i) = f_L(w_i)$.
Now let $x \notin L$. Thus we will never halt on step 1.c and reach after n iteration step 2 in which we will halt with the value 0.
Making $f_M(x) = 0$.
In conclusion, for a given L finite language, we can build M TM s.t. $\forall x \in \Sigma^* : f_M(x) = f_L(x)$. Thus f_L is computable.
Q.E.D $\square$