HW2CSDS343

1

> **Question**
>
> Let L_1 and L_2 be two languages over Σ. Prove that if L_1, L_2 are both Turing-decidable then $L_1 \oplus L_2$ is Turing-decidable.

Assume L_1, L_2 are Turing decidable.
$\exists M_1, M_2$ that decide L_1, L_2
Create M_3
M_3 runs on string x:

- Duplicate x after a #
- Run M_1 on x
  - If M_1 accepts:
    - Clear everything after #
    - Duplicate x after #
    - Run $M_2 on\$x$
    - If M_2 accepts:
      - Output "No"
    - Else:
      - Output "Yes"
  - Else:
    - Clear everything after #
    - Duplicate x after #
    - Run M_2 on x
      - If M_2 accepts:
        - Output "Yes"
      - Else:
        - Output "No"

> **Proof**
>
> Show M_3 decides $L_1 \oplus L_2$
> If $x \in (L_1 \oplus L_2$ then $(x \in L_1 \wedge x \notin L_2) \vee (x \notin L_1 \wedge x \in L_2)$

- If $x \in L_1 \wedge x \notin L_2$ - If $x \in L_1 \wedge x \not inL_2$
  - M_3 will run M_1 which accepts, then it will run M_2 which rejects. So M_3 accepts.
- If $x \notin L_1 \wedge x \in L_2$
  - M_3 will run M_1 which rejects, then it will run M_2 which accepts. So M_3 accepts.

If $x \notin L$ then $(x \in L_1 \wedge x \in L_2) \vee (x \notin L_1 \wedge x \notin L_2)$

- If $x \in L_1 \wedge x \in L_2$
  - M_3 will run M_1 which accepts, then it will run M_2 which accepts. So M_3 rejects.
- If $x \notin L_1 \wedge x \notin L_2$
  - M_3 will run M_1 which rejects, then it will run M_2 which rejects. So M_3 rejects.

2

Question

Let L_1, L_2 be 2 languages over Σ. Prove that is L_1, L_2 are both Turing decidable then the concatenation of L_1, L_2 is Turing decidable.

Assume L_1, L_2 are Turing decidable.
$\exists M_1, M_2$ that decide L_1, L_2
Create M_3
M_3 runs on string x:

- let n = length(x)
- insert # after x on tape
- insert a # after #
- loop 1:
  - for each 0 between the #'s copy a character at the front of x to after the 2nd #
    - Run M_1 on string after #
    - If M_1 accepts:
      - Clear after the #
      - Copy all unmarked characters of x (copy suffix)

    - Run M_2 on string after #
    - If M_2 accepts:
      - Output "yes"
    - Else:
      - Continue
  - Else:
    - Continue
  - Clear after the #
  - If number of 0's greater than length of x
    - Output "no"
  - Else:
    - Add a new 0 between the #'s
    - Go back to loop 1

In general, the tape is structured as follows: $x\#i\#$copy, where i is length of prefix

Proof

Show M_3 decides the concatenation of L_1, L_2
If $x \in$ concatenation of $L_1, L2$, then $\exists i \in \mathbf{Z}$ where $0 \leq i \leq length(x)$ such that $x_{1,i} \in L_1 \wedge x_{i+1,length(x)} \in L_2$

We know the above statement is true by the definition of the concatenation of $L_1, L2$.
Note that given a substring $x_{j,k}$ if $j > k$ the string is invalid and thus blank. Also bounds are inclusive.

M_3 iterates through all possible values of i $(0, 1, 2, \ldots, length(x))$
On any iteration $x_{1,i} \notin L_1, M_1$ will reject so M_3 continues.
If $x_{1,i} \in L_1$ then M_1 will accept so M_3 runs M_2 on $x_{i+1,length(x)}$.
If M_2 accepts, M_3 accepts, otherwise M_3 continues.
If either M_2 or M_1 reject we go to the next prefix / suffix pair.
As such, we exhaust every prefix / suffix pair.
Therefore, if $x \in$ the concatenation of L_1, L_2, M_3 will find the valid prefix / suffix pair through exhaustion.

If $x \notin$ the concatenation of L_1, L_2, then $\nexists$ a prefix / suffix pair s.t. prefix $\in L_1 \wedge$ suffix $\in L_2$
At each iteration, test a possible suffix / prefix. M_1 will reject the prefix so we continue to the next prefix.

If prefix $\in L_1, M_1$ will accept but we know the suffix is not in L_2 so M_2 will reject
After we have tested the entire length of the string M_3 will reject.

3

> **Question**
>
> Let L_1, L_2 be 2 languages over Σ. Prove that is L_1, L_2 are both Turing recognizable then the concatenation of L_1, L_2 is Turing recognizable.

Assume L_1, L_2 are Turing recognizable.
$\exists M_1, M_2$ that decide L_1, L_2
Create M_3 (2 way infinite Tape)

- let n = length(x)
- insert # after x on tape
- insert # after the #
- insert # before x on tape
- loop 1:
    - insert one 0 to the left of the leftmost #
    - loop 2:
        - for each 0 between the #'s to the right of x, copy a character at the front of x to after the rightmost #
        - for each 0 to the left of x run a step of M_1 on the copied string
        - If M_1 accepts:
            - Clear after the #
            - Copy all unmarked characters of x (copy suffix)
            - Run M_2 on string after # for i steps
            - If M_2 accepts:
                - Output "Yes"
            - Else:
                - Continue
        - Else:
            - Continue
        - Clear after #
        - If number of 0's greater than length of x
            - Continue

- Else
    - Add a new 0 in between the #'s to the right of x
    - Go back to loop 2
- Go back to loop 1

In general, the tape is structured as follows: $i\#x\#j\#$copy, where i is the number of steps and j is length of prefix

Proof

If $x \in$ the concatenation of L_1, L_2, then $\exists$ a prefix / suffix pair s.t. prefix $\in L_1 \wedge$ suffix $\in L_2$
If prefix $\in L_1$ then M_1 will halt and accept in a finite number of steps.
A similar assertion can be made for M_2 recognizing the suffix.
M_3 runs M_1 on the prefix for i finite steps.
In the case $x \in$ the concatenation, then M_1 will recognize some prefix in finite number of steps.
Simlarly, once the prefix is recognized, M_3 runs M_2 on the suffix.
Since L_2 is Turing recognizable, M_2 will recognize the suffix in finite number of steps.

If $x \notin$ the concatenation, then $\nexists$ a prefix / suffix pair s.t. prefix $\in L_1 \wedge$ suffix $\in L_2$
As such for any prefix, M_1 will either reject or run forever
Similarly, M_2 will either reject or run forever
As such, M_3 will run forever.