CS 39: Theory of Computation **Winter '23**

PSET 6 — 02/27/2023

Prof. Chakrabarti ***Student: Amittai Siavava***

Credit Statement

I discussed ideas for this homework assignment with Paul Shin.

I also referred to the following books:

(a) **Introduction to the Theory of Computation** by **Michael Sipser**.

Problem 1.

We remarked in class that decidable languages are closed under (a) union, (b) intersection, (c) complement, (d) concatenation, and (e) Kleene star. Prove the following.

(b) intersection

Let T_1 and T_2 be TMs deciding L_1 and L_2 respectively. Construct a TM T deciding $L_1 \cap L_2$ as follows:

```
T = "On input $w$:
  1. Simulate $T_1$ on input $w$.
  2. If $T_1$ rejects, REJECT .
  3. If $T_1$ accepts, simulate $T_2$ on input $w$.
  4. If $T_2$ accepts, ACCEPT ; if $T_2$ rejects, REJECT ."
```

Since T_1 and T_2 are both deciders, T is always terminates with either ACCEPT or REJECT , and it only returns ACCEPT when both T_1 and T_2 accept the input so it is a decider for $L_1 \cap L_2$. □

(c) complement

Let T be a TM deciding L. Construct a TM T' as follows:

```
$T'$ = “On input $w$:
  1. Simulate $T$ on input $w$.
  2. If $T$ accepts, Reject ; if $T$ rejects, Accept .”
```

Since T is a decider, T' always terminates on any input string. Furthermore, T' always disagrees with T on any string — if T' accepts w, then T rejects w and vice versa, so T is a decider for $\overline{L}$. □

(e) Kleene star

Let T be a TM deciding L. Construct an NDTM T' as follows:

```
$T'$ = “On input $w$:
  1. If $w = \varepsilon$, Accept .
  2. Insert a new symbol $\# \notin \Sigma$ before the first symbol of $w$.
  3. Simulate $T$ on the contents of $w$ before the $\#$ symbol.
  4. If $T$ rejects, then; if the the $\#$ symbol is not at the end of $w$, shift it by 1 to the right and go to step 3.
  5. If $T$ accepts, non-deterministically choose one of the following two options:
      5.1. Simulate $T'$ on the contents of $w$ after the $\#$ symbol. If $T'$ accepts, Accept .
      5.2. Shift the $\#$ symbol by 1 to the right and go to step ??
```

T' accepts an input w under any of these scenarios:

(i) $w = \varepsilon$ or w is a member of L.

(ii) w contains a prefix that is a member of L, and the remaining suffix is also accepted by T'.

Therefore, if T' accepts w, then (a) w is empty or (b) by unwinding the recursion we can write $w = w_1, w_2, \dots, w_n$ where $w_i \in L$ for all i. This is the definition of w being in L^*, so T' is a decider for L^*. □

Problem 2.

Furthermore, recognizable languages are closed under (a) union, (b) intersection, (c) concatenation, (d) Kleene star, and (e) the HALF operation. Prove the following:

(a) union

Let T_1 and T_2 be TMs recognizing L_1 and L_2 respectively. Construct a 2-tape TM T as follows:

T = "On input w:

1. Copy w onto the second tape.
2. Simulate T_1 on input w on tape 1 and concurrently simulate T_2 on input w on the second tape.
3. If T_1 halts and accepts, ACCEPT .
4. If T_2 halts and accepts, ACCEPT .
5. Otherwise, keep running T_1 and T_2."

If $w \in L_1 \cup L_2$, then either T_1 will eventually halt and accept or T_2 will eventually halt and accept, so T will eventually halt and accept. However, if $w \notin L_1 \cup L_2$, then T_1 and t_2 may never halt, so T may also never halt. Therefore, T recognizes, but does not decide, $L_1 \cup L_2$. □

(c) concatenation

Let T_1 and T_2 be TMs recognizing L_1 and L_2 respectively. Construct an NDTM T as follows:

```
$T$ = "On input $w$:
  1. Insert a new symbol $\# \notin \Sigma$ before the first symbol of $w$.
  2. Non-deterministically do one of the following:
      2.1. Simulate $T_1$ on the contents of $w$ before the $\#$ symbol and simulate $T_2$ on the contents of $w$
           after the $\#$ symbol. If both accept, Accept .
      2.2. Shift the $\#$ symbol to the right by 1 and restart step 2."
```

If T accepts a string w, then there must exist some prefix of w accepted by T_1 such that the remainder of w is accepted by T_2. Therefore, $w \in L_1 L_2$. Similarly, if $w \in L_1 L_2$, then it can be split into two strings w_1 and w_2 such that $w_1 \in L_1$ and $w_2 \in L_2$, so T will accept w.

However, since T_1 and T_2 may not halt on inputs not in L_1 and L_2 respectively, T may not halt on inputs not in $L_1 L_2$, so it does not decide $L_1 L_2$. □

(e) HALF

Let L be a language over an alphabet Σ, and let T be a TM recognizing L.

First, construct an enumerator TM N that enumerates members of Σ of a specific length.

```
N = "On input $n$:
  1. $l \leftarrow 0$.
  2. while $l$ <= $n$:
       2.1. Determine the next member $x \in \Sigma^*$ (in lexicographic ordering).
       2.2. If $|x| = n$, LIST $x$.
       2.3. $l \leftarrow |x|$."
```

To recognize HALF (L), we construct a TM T' as follows:

```
$T'$ = "On input $w$:
  1. $n \leftarrow |w|$.
  2. Simulate $N$ on input $n$.
  3. For each $x \in \Sigma^*$ enumerated by $N$:
       3.1. Simulate $T$ on $wx$.
       3.2. If $T$ accepts, ACCEPT .
```

Problem 3.

Prove that every infinite recognizable language has an infinite decidable subset. (If L is an infinite recognizable subset, then there exists an infinite decidable language $L' \subseteq L$).

Hint: Think of enumerator TMs and the results we proved in class about them.

Let M be a TM recognizing a language L over alphabet Σ.

Note: I use the fact that Σ^* is decidable thus enumerable in lexicographic order to enumerate members of Σ^* without going into the full details of the TM for this operation since we covered that in class.

First, construct an enumerator TM E for a subset of the language L as follows:

```
$E$ = “On input $\langle M \rangle$:
    1. $len \leftarrow 0$.
    2. For $i \leftarrow 1, 2, 3, \ldots$
        2.1. Define $s_i$ to be the next string in the enumeration of $\Sigma^*$.
        2.2. Define $S = \{s \in \{s_1, s_2, \ldots s_i\} : |s| > len\}$.
        2.3. Simulate $M$ for $i$ steps on each $s \in S$.
        2.4. If $M$ accepts some $s_k \in S$
            2.4.1. LIST $s_k$
            2.4.2. $len \leftarrow |s_k|$”
```

Claim 3.1. *There exists a language $L' \subseteq L$ such that L' is infinite and decidable.*

Proof. Let $L' = \{w \in L : w \text{ is listed by } E\}$.

First, note that L is infinite, so L' also has to be infinite since, for any length l, E can always list another string $s \in \Sigma^*$ of length greater than l. Additionally, Σ^* is enumerated in lexicographic order, so the sequence of strings listed by E does not change across different runs of E. Thus, L' is decidable by the following TM:

```
$T$ = “On input $w$:
    1. Simulate $E$ on input $\langle M \rangle$.    (where $M$ is the original TM for $L$)
    2. If $E$ lists $w$, ACCEPT .
    3. The moment $E$ lists a string $w'$ such that $|w'| \geq |w|$, REJECT .
```

T will always terminate with either ACCEPT or REJECT . □

Problem 4.

For each of the following languages, classify the language into one of the following categories: (a) unrecognizable; (b) recognizable, but not decidable; (c) decidable.

(a) $L = \{\langle M\rangle : M$ is a TM and M at least two strings.$\}$

L is recognizable but not decidable.

(b) $L = \{\langle M\rangle : M$ is a TM and M accepts exactly two strings.$\}$

(c) $L = \left\{\langle M_1, M_2\rangle : M_1\right.$ and M_2 are TMs over the input alphabet $\{0,1\}$ and $\left.\mathcal{L}(M_1) = \{0,1\}^* - \mathcal{L}(M_2).\right\}$

Problem 5.

All of the automata models we have studied in this course allow "useless" states, i.e., states which are never entered in any run. It would be nice to have an algorithm that could detect and prune such useless states in an automaton, but this is not always possible! Define the languages

$$\mathsf{U}_{\mathsf{DFA}} = \{\langle M \rangle : M \text{ is a DFA and } M \text{ has at least one useless state}\}\,,$$

$$\mathsf{U}_{\mathsf{PDA}} = \{\langle M \rangle : M \text{ is a PDA and } M \text{ has at least one useless state}\}\,,$$

$$\mathsf{U}_{\mathsf{TM}} = \{\langle M \rangle : M \text{ is a TM and } M \text{ has at least one useless state}\}\,.$$

(a) Prove that $\mathsf{U}_{\mathsf{DFA}}$ is decidable.

(b) Prove that $\mathsf{U}_{\mathsf{PDA}}$ is decidable.

Problem 6.

All of the automata models we have studied in this course allow “useless” states, i.e., states which are never entered in any run. It would be nice to have an algorithm that could detect and prune such useless states in an automaton, but this is not always possible! Define the languages

$$\mathsf{U}_{\mathsf{DFA}} = \{\langle M\rangle : M \text{ is a DFA and } M \text{ has at least one useless state}\},$$

$$\mathsf{U}_{\mathsf{PDA}} = \{\langle M\rangle : M \text{ is a PDA and } M \text{ has at least one useless state}\},$$

$$\mathsf{U}_{\mathsf{TM}} = \{\langle M\rangle : M \text{ is a TM and } M \text{ has at least one useless state}\}.$$

Prove that U_{TM} is undecidable.