Max Shi
CS 334
Professor Bhatt
November 5, 2019
I pledge my honor that I have abided by the Stevens Honor System.

Problem Set 7

1. **Union** – Let there be a new TM D, where the input be two TMs, A and B, which decides $L_a$ and $L_b$, and a string w. Let the TM D run w on both TMs A and B. If either A or B accepts w, let D accept w. Otherwise, reject. Thus, D accepts w if w ∈ (A ∪ B) and rejects if w ∉ (A ∪ B), thus D decides (A ∪ B).
   **Concatenation** – Let there be a new TM D, where the input is two TMs, A and B, which decides $L_a$ and $L_{b,}$ and a string w. Let the TM non-deterministically partition the string w into two distinct parts, a and b, where a is before b and where 0 <= |a| <= |w| and |a|+|b| = |w|, allowing both a and b to be empty strings or the entirety of w. Then, run the two partitions a and b on A and B, respectively, and accept w if both A and B accept any pair of a and b or reject if this never happens. Thus, D will accept all strings w if it is a string from $L_a$ and a string from $L_b$ are concatenated, and reject otherwise.
   **Star** – Let there be a new TM D, where the input is one TM A that decides $L_a$ and an input string w. Accept if w is the empty string. Otherwise, non-deterministically partition w into at least 1 part and at most |w| parts, where for each part p, 0 < |p| <= |w|. Then, run A on each partitioned piece of the string w. If A accepts each piece of the string w from one set of partitions, accept, otherwise, if this never happens, reject. Thus, D decides strings in the language $L_a$*.
   **Intersection** – Let there be a new TM D, where the input be two TMs, A and B, which decides $L_a$ and $L_b$ and a string w. Let the TM D run w on both TMs A and B. If both A and B accepts w, let D accept w, otherwise reject. Thus, D accepts w if w ∈ (A ∩ B), and rejects if w ∉ (A ∩ B), thus D decides (A ∩ B).
   **Complement** – Let there be a new TM D, where the input is one TM A which decides $L_a$ and an input string w. Then, make D run w on A. If A rejects, let D accept, and if A accepts, let D reject. Thus, D decides the complement of $L_a$, as D will reject all strings in the language and D will accept all strings not in the language.
   **For all of these, because we have a TM that decides each of these operations, TM-decidable languages are closed under these operations.**
2. **Union** – Let there be a new TM D, where the input be two TMs, A and B, which recognizes $L_a$ and $L_b$, and a string w. Let the TM D run w on both TMs A and B non-deterministically to account for the machine never halting. If either A or B accepts w, let D accept w. Thus, D accepts w if w ∈ (A ∪ B), and either rejects or never halts if w ∉ (A ∪ B), thus D recognizes (A ∪ B).
   **Concatenation** – Let there be a new TM D, where the input is two TMs, A and B, which recognize $L_a$ and $L_b$, with a string w. Let the TM non-deterministically partition the string w into two distinct parts, a and b, where a is before b and where 0 <= |a| <= |w| and |a|+|b| = |w|, allowing both a and b to be empty strings or the entirety of w. Then, run the two partitions a and b on A and B, respectively, and accept w if both A and B accept any pair of a and b or reject if this never happens. Thus, D will accept all strings w if it is a string from $L_a$ and a string from $L_b$

are concatenated, and not accept otherwise, thus recognizing concatenation of $L_a$ and $L_b$.

**Star** - Let there be a new TM D, where the input is one TM A that recognizes $L_a$ and an input string w. Accept if w is the empty string. Otherwise, non-deterministically partition w into at least 1 part and at most |w| parts, where for each part p, 0 < |p| <= |w|. Then, run A on each partitioned piece of the string w. If A accepts each piece of the string w from one set of partitions, accept, otherwise, reject or run forever if A never halts on some partition. Thus, D recognizes strings in the language $L_a$*.

**Intersection** - Let there be a new TM D, where the input be two TMs, A and B, which recognizes $L_a$ and $L_b$ and a string w. Let the TM D run w on both TMs A and B. If both A and B accepts w, let D accept w, otherwise run on forever if one does not halt or reject if both reject. Thus, D accepts w if w ∈ (A ∩ B), and either rejects or runs forever if w ∉ (A ∩ B), thus D recognizes (A ∩ B).

**For all of these, because we have a TM that recognizes each of these operations, TM-recognizable languages are closed under these operations.**

**Complement** – Let there be a TM D that takes in as input a TM A that recognizes language L and an input string w, that recognizes the complement of L. However, if this language is TM-recognizable, this means that there can exist some string s that causes A to run forever, and never accept nor reject. Thus, it is impossible to say whether or not A will accept or reject this string s. Thus, it is then impossible for TM D to do the opposite of what TM A will do. Thus, this creates a contradiction, as D cannot provide the opposite of TM A for this string s, and D cannot recognize the complement of language L.

3. **Proving the iff from right to left,** we need to prove that the existence of a lexicographic enumerator allows us to create a TM that decides the language. Let this TM D have in its input a lexicographic enumerator for the language L and an input string w. Let D begin reading strings in the lexicographic enumerator. If it finds w in the enumerator, then it accepts, as w was in the language. If D gets to the point in the enumerator where it reads a string s with |s| > |w|, then we know that w was not a part of the language L and D can reject. This cannot run on forever, as w has finite length and there are a finite amount of strings of each length. Thus, D can decide if any string w is in L or not, thus proving that L is a TM-decidable language.

   **Proving the iff from left to right,** we need to prove that a language being decidable allows us to create a lexicographic enumerator. Start with a TM D that takes as input a TM A that decides a TM-decidable language L. Let D take strings from the enumerator of all possible strings, which prints in standard order (alphabetically and in increasing length). For each string read, one at a time, run TM A on each string from the enumerator, and print them out if A accepts the string. Because A decides if w is in the language, A will always halt, and we can retain the same standard order of the original enumerator. Thus, D becomes a lexicographic enumerator for the language L, as it will print out strings in increasing length order that are in the language L.

4. Let L be an infinite TM-recognizable language, and let A be the TM that accepts L. Thus, let E be an enumerator for every string that A accepts. Because L is an infinite language, E will continue to print out strings that A accepts forever. Then, let C be a TM that utilizes the enumerator S of all possible strings, and takes TM A as input. Then, let C run A on each string that S prints out. Then, let C print the string from S if A rejects the string. Thus, C becomes an enumerator for all strings that A rejects. Again, because L is an infinite TM-recognizable language, C will continue to print out strings forever. Thus, unioning the output of these two enumerators, E and C, will create a subset of all strings that can be put into A, but because all strings printed by E or C have

either been accepted or rejected by A, this subset E ∪ C defines the decidable subset of A. Furthermore, because both E and C will continue to print strings forever, this subset is also infinite.