Final pract sol

Introduction to Theoretical Computer Science (Concordia University)

CSCI 2400 – Models of Computation

Solutions to Practice Final Exam

Here are solutions to the practice final exam. For some problems some details are missing for brevity. You should write complete solutions at the final exam.

1. Prove that the following language is not context-free.

$$L = \{a^i b^j c^k : 0 \leq i \leq j \leq k\}$$

Answer. Let m be the parameter of the pumping Lemma. We choose to pump the string $a^m b^m c^m$.

The string vxy has length at most m. Notice that v cannot be simultaneously in a^m and b^m, since if we repeat v (we take v^2xy^2) then a's are mixed with b's and the resulting string is not in the language L. Similarly, v cannot be simultaneously in b^m and c^m. A similar observation holds for y.

Now let's consider the case where vxy is completely within a^m. If we pump up the string vxy and we take v^2xy^2, then the resulting string has the form $a^{m+k}b^m c^m$, for some $k > 0$ (since $|vy| \geq 1$), which is not in the language L. For similar reasons vxy cannot be completely within b^m and c^m.

Now we consider the case where vxy is such that v is completely within a^m, and y is completely within b^m. If pump up the string vxy once and we obtain the string v^2xy^2, then the resulting string has the form $a^{m+k_1}b^{m+k_2}c^m$, with $k_1 + k_2 \geq 1$ (since $|vy| \geq 1$), which is not in the language L. We treat similarly the case where vxy is such that v is completely within b^m, and y is completely within c^m.

Therefore, in all cases the resulting string is not in the language L, and therefore the language L is not context-free.

2. Prove that the following language is context-free.

$$L = \{ww^R : w \neq abba, \text{ and } w \in \{a, b\}^*\}$$

Answer. We know that the language $L_1 = \{ww^R : w \in \{a, b\}^*\}$ is context-free, since it is generated by the context-free grammar:

$$S \to aSa|bSb|\lambda$$

The language $L_2 = \{abbaabba\}$ is regular, since it is described by the regular expression $abbaabba$. Therefore, the language $\overline{L_2}$ is regular too (the complement of a regular language is regular). We have:

$$L = L_1 \cap \overline{L}_2$$

From Theorem 8.5, page 223, we have that the intersection of a context-free language with a regular language is context-free. Therefore, the language L is context-free.

3. Design a Turing Machine that accepts the following language.

$$L = \{ab(a+b)^*\}$$

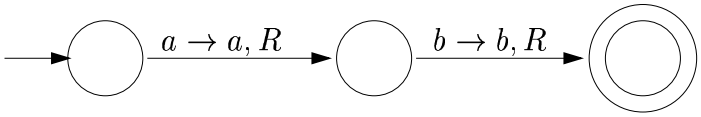

Figure 1: Turing machine for language $L = \{ab(a+b)^*\}$

Answer. The Turing machine for language L is shown in Figure 1. Notice that the machine doesn't have to read all the input string in order to accept the string. This is because we know the input string alphabet which is $\{a, b\}$, and after string ab any string made from the input alphabet can follow.

4. Describe the algorithm of a three-tape Turing machine that computes the following function.

$$f(x) = x^2$$

Answer. The algorithm is the following:

`Compute-Square`

Input: The input is number x in unary notation which appears on the first tape.

i. Copy x to the second and third tape.

ii. Append the contents of the second tape to the end of the contents of the first tape.

iii. Remove one 1 from the string of the third tape.

iv. If more than one 1's are still on the third tape goto step ii.

5. Show that the class of Pushdown Automata with two stacks are equivalent with the Turing machines.

Answer. We want to show that Turing machines can simulate Pushdown Automata with two stacks and vice versa.

- **Turing machines simulate pushdown automata with two stacks.**

  Pushdown automata are non deterministic. For this reason we choose non-deterministic Turing machines (which are equivalent to standard Turing machines) to simulate the pushdown automata.

  We split the tape of the Turing machine into two halfs. We place a marker symbol somewhere on the tape to denote the border between the two halfs. The part of the tape left from the marker simulates one stack and the part of the tape right from the marker simulates the second stack. The bottom of the two stacks is the marker. You can imagine the rest of the details.

- **Pushdown automata with two stacks simulate Turing machines.**

  The only difficultly is to simulate the Turing machine tape with two stacks. We can think of the two stacks as being concatenated at their tops. When we want to get to a symbol which is in the first stack then we pop all the contents of the first stack above that symbol and we push these contents to the second stack. This way we can access any symbol stored in a stack without loosing any contents. Essentially this simulates the tape of a Turing machine where the head corresponds to the heads of the two stacks (which point at the top of the stacks). You can imagine the rest of the details.

6. Describe a procedure that enumerates the following strings in proper order.

$$S = \{1^{2^n} : n \geq 0\}$$

Answer. The proper order is:

$$1, 11, 1111, 11111111, \ldots$$

Notice that each string is twice as big as the previous string. The enumeration procedure is the following.

`Enumeration Procedure`

i. Write 1 on the tape. This is an enumerated string.

ii. Copy the contents of the previously enumerated string next to that string. This is an enumerated string.

iii. Goto step ii.

7. Prove that the family of recursively enumerable languages is closed under union.

Answer. Let L_1 and L_2 be two recursively enumerable languages. We want to prove that the language $L = L_1 \cup L_2$ is recursively enumerable. We can prove this by giving an enumeration procedure for L (see the discussion on page 287). Since L_1 and L_2 are recursively enumerable then there are respective enumeration procedures M_1 and M_2 for each language. We can construct an enumeration procedure M for L as follows. Enumeration procedure M interleaves the computation of M_1 and M_2. In particular, M allows M_1 to write its first string, then allows M_2 to write its first string, then allows M_1 to write its second string, then allows M_2 to write its second string, and so on. The above procedure enumerates all the strings of L.

As an alternative proof you could construct a Turing machine that accepts the union.

8. Prove that the family of recursive languages are closed under intersection.

Answer. Let L_1 and L_2 be two recursive languages. We want to prove that the language $L = L_1 \cap L_2$ is recursive. Since L_1 and L_2 are recursive, there are respective Turing machines M_1 and M_2 which accept each language and halt for each input string. From M_1 and M_2 we can construct a Turing machine that accepts L and halts for each input string as follows. For input string w machine M first allows machine M_1 to run with input string w. If M_1 rejects then M rejects. If M_1 accepts then M allows M_2 to run with input string w. If M_2 accepts then M accepts. If M_2 rejects then M rejects.

9. Prove that the problem of determining whether $w \in L(G)$, where G is a context-free grammar, is decidable.

Answer. This problem is decidable because there are many algorithms that answer this problem. See for example the exhaustive search parsing method, described in page 141, and the CYK algorithm, described in Section 6.3, page 178.

10. Prove that the problem of determining whether two Turing machines accept the same language is undecidable.

Answer. This problem is equivalent with the following problem. Determine for two recursively enumerable languages L_1 and L_2 whether $L_1 = L_2$. We will call this the "language-equality problem".

We know that the following problem is undecidable. Given any recursively enumerable language L_1 determine whether $L_1 = \emptyset$ (see Theorem 12.3, page 322). We will call this the "empty-language problem".

We will reduce the empty-language problem to the language-equality problem. In particular, set $L_2 = \emptyset$ (trivially, L_2 is a recursively enumerable language since we can easily construct a Turing machine that accepts no string). Obviously, if we can solve the language-equality problem $L_1 = \emptyset$, then we can solve the empty-language problem $L_1 = \emptyset$.