CS 5000 Homework 6

Sally Devitry (A01980316)

Problem 1

Let $L = \{a^n b^n c^n \mid n \geq 0\}$. We've proved before that L is not context-free. Sketch a Turing Machine (TM) for accepting L. When I say sketch I don't mean that you have to give the exact quadruples of your TM. Instead I want you to describe the workings of your TM in statements similar to the statements below.
1. If the current symbol is B, go to step i;
2. If the current symbol is a, print X over it and go to step j;
3. Move right past any a's and Y 's until the first b occurs;
4. Move left past any a's, b's, Z's, and Y's until the first X;
5. Halt.

Solution

1. Mark 'a' as 'X' and move right until an unmarked 'b' is reached.
2. Mark 'b' as 'Y' and move right until an unmarked 'c' is reached.
3. Mark 'c' as 'Z' and move left until 'X' is reached.
4. Move to 'a' and repeat steps 1-4.
5. If the tape head reaches 'X' and the next symbol is 'Y' that means 'a's are finished.
6. Move right until the end of the tape (BLANK is reached). If no 'b's or 'c's are found, then the string is accepted.
7. Halt

Problem 2

Imagine a TM with two separate tapes: tape 1 and tape 2. Each tape has a separate reading head that can move one cell left or right at a time, like a standard TM with one tape. At the beginning of the computation, the reading heads are aligned and are reading the blank (B) in the cell to the left of the input. The transition function for this TM takes three inputs: the current state, a symbol on tape 1, and a symbol on tape 2. On each tape the TM can print a new symbol or move the tape's reading head one cell left or right, after which it can go to a new state or remain in the current state. Explain how the finite control tuples can be defined for a 2-tape TM. Explain how a 1-tape TM can simulate any computation performed by a 2-tape TM. Basically, your explanation is a proof sketch that multiple tapes don't buy us any additional recognition power.

Solution

At any given moment, the 2-tapes in the described Turing Machine only have a finite amount of information on them, so you can write that all down on a single tape with separators to define the contents of different tapes, and dots to indicate where the 2 tape-heads are located. The finite control tuples could be in the form, $\#l_1, w\#l_2, w$, where l_k is the location of the kth tape head, and w_k, is the content at that location. This means that a 2-tape machine and a 1-tape machine are equivalent and a 2-tape machine buys no recognition power.

Problem 3

Imagine the tape is replaced with a 3D cube extending infinitely from an arbitrary origin at (0, 0, 0). The input to the TM is written in the cells (0, 0, 1), (0, 0, 2), (0, 0, 3), ..., (0, 0, k). Explain how the quadruples for this 3D TM can be defined. Can a 3D TM be simulated by a 1-tape TM? If yes, explain how. If no, explain why not.

Solution

Yes, a 3D Turing Machine can be simulated by a 1-tape Turing Machine because at any given time the 3-tapes of a 3D TM have a finite amount of information contained on them. The quadruples for this TM can be defined in the form, $\#x, y, z, v\#$, where x, y, z are coordinates, v is the value at the location that the coordinates point to and the # is a separator.

Problem 4

Explain how a TM can simulate an arbitrary DFA. Hint: Use a 2-tape TM. You can do it with a 1-tape TM, too, but a 2-tape TM will likely make your explanation much more intuitive.

Solution

A Turing Machine can easily simulate an arbitrary DFA. Each transition in a DFA reads a character, follows a transition, then moves to the next input. After all input is read, the DFA accepts if it has reached an accepting state.
To simulate a Turing Machine, You can build the finite state control of the Turing Machine by creating one state for each state in the DFA. For each transition in the DFA, on any character, it should write back an arbitrary character specific to that seen character. The tape head moves right (another state in the DFA) after the arbitrary character is written.
Then, based on whether a state is accepting or rejecting, define a transition for each state on the blank symbol from that state either to the accept or reject state of the Turing Machine.
The described Turing machine simulates a DFA by stepping manually across the input string and finally deciding whether the end state is accepting or rejecting.

Problem 5

Consider the following L-program P.

```
[A1] IF X1 != 0 GOTO B1
Z1 ← Z1 + 1
IF Z1 != 0 GOTO E1
[B1] X1 ← X1 - 1
Y ← Y + 1
Z1 ← Z1 + 1
IF Z1 !=0 GOTO A1
```

Write a computation of P that starts with the snapshot (1,σ), where $\sigma = \{X1 = 2, Y = 0, Z1 = 0\}$.

Solution

Following are the snapshots from the program, P.
$(1, \{X1 = 2, Y = 0, Z1 = 0\})$
$(2, \{X1 = 1, Y = 1, Z1 = 1\})$
$(3, \{X1 = 0, Y = 2, Z1 = 2\})$
$(4, \{X1 = 0, Y = 2, Z1 = 3\})$

Problem 6

Write a program in L such that for every computation $s_1, ..., s_k$ of P, $k = 3$.

Solution

A program where k always equals 3, means that there are always exactly 3 snapshots.
The following program satisfies the condition.
$X \leftarrow X$
$X \leftarrow X$
$X \leftarrow X$