**3.1.a**

**3.1.c**

**3.1.d**

**3.2.b**

**3.2.c**

**3.7**

In step (1) ”try all possible settings”, there are infinite settings, which makes it an illegitimate Turing machine description.

**3.8.c 1**

“On input string w:

1. Scan the tape and mark the first 0 which has not been marked. If there is no unmarked 0, go to stage 5.

2. Move on and mark the next unmarked 0. If there is not any on the tape, accept. Otherwise, move the head back to the front of the tape.

3. Scan the tape and mark the first 1 which has not been marked. If there is no unmarked 1, accept.

4. Move the head back to the front of the tape and repeat stage 1.

5. Move the head back to the front of the tape. Scan the tape to see if there are any unmarked 1s. If there is not, reject. Otherwise, accept.”

**3.9.a**

(1) All languages that can be recognized by 1-PDA can also be recognized by a 2-PDA.

(2) Some non-context-free languages can also be recognized by a 2-PDA.

It has been shown that is not context free; in the other word we cannot construct a 1-PDA that recognizes L. However, we can construct a 2-PDA M in the following way such that it can accept L.

1, When M reads in an ‘a’, it pushes ‘a’ onto stacks 1.

2, After M begins to read in ‘b’, no more ‘a’ is allowed in the input.

3, When M reads in a ‘b’, it pushes ‘b’ onto stack 2.

4, After M begins to read in ‘c’, no more ‘a’ or ‘b’ is allowed in the input

5, When M reads in a ‘c’, it pops an ‘a’ from stack 1 and a ‘b’ from stack 2.

6, Accept If and only if by the end of the input, both of the stacks are empty

(1) and (2) show that 2-PDA is more powerful than 1-PDA.

**3.9.b**

(1) The power of the 2-PDAs is equivalent to the standard Turing Machines.

(1.1) we can construct a 2-PDA M that recognizes the same language as a Turing Machine TM.

The tape of a TM can be split into two: left and right, and each are stored in a stack. Stack R stores the character on the right, from the current pointer position to the right; and Stack L stores the character on the left, from current position to the left.

For each transition in TM, the M pops off Stack L, pushes onto Stack R, and goes from state to .

For each transition in TM, the M pops off Stack R, pushes onto Stack L, and goes from state to .

(1.2) we can easily replace the two stacks of 2-PDA with 2 tapes and construct a 2-Tape TM. And 2-Tape TM is as powerful as 1-Tape TM (theorem 3.8).

(2) A 3-tape TM is more powerful than a 3-PDA.

The construct above showed that one tape can be simulated with 2 stacks. One tape that can be read-and-write, left-and right , is more powerful than a stack.

(3) A multi-tape TM is not more powerful than a standard TM (theorem 3.8)

(1),(2),(3) together show 3-PDAs are not more powerful than 2-PDAs.

**3.16.c 2**

For any Turing-recognizable language L, let M be the TM that recognizes it. We construct a Non-deterministic 2-tape TM M0 that recognizes the star of L:

”On input w:

1, If tape 1 (of M) has only a blank, accept.

2, Scan the input tape (tape 1) from left to right until a blank is encountered. For each tape cell read, nondeterministically choose either to write the same symbol, or to write the symbol with a mark on it. Any number of tape cells can be marked in this process. Position the tape head on the first tape at the left hand end.

3, While the symbol on the first tape head under the tape head is not a blank, go to step 4. Otherwise move the header of the first tape to the beginning and go to step 2 (Guess a different separation).

4, Make the second tape entirely blank. (Start at the left hand end, and move right writing blanks until the first blank is read.)

5, Copy all symbols from the first tape starting at the current position of the tape head up to and including the marked symbol onto the second tape (starting at the left hand end of the second tape), removing the mark on the last symbol. Leave the tape head on the first tape positioned on the cell right after the last (marked) symbol copied.

6, Run M on the contents of the second tape. If it rejects, then M0 rejects and stops, if it halts, then M0 halts, and if it accepts, go to step 3.”

Nondeterminism is used to break the string up into some number of pieces. The second tape is introduced to preserve the string so that we can run many trials.

**3.16.d**

For any two Turing-recognizable languages L1 and L2, let M1 and M2 be the TMs that recognize them. We construct a TM M0 that recognizes the intersection of L1 and L2:

“On input w:

1. Run M1 on w, if it halts and rejects, reject. If it accepts, go to stage 2.

2. Run M2 on w, if it halts and rejects, reject. If it accepts accept."

If both of M1 and M2 accept w, w belongs to the intersection of L1 and L2 and M0 will accept w after a finite number of steps.

**3.21**

1) if , then

2) if ,

In conclusion,

References:

1, <http://cc-li.blog.ntu.edu.tw/files/2010/12/2010hw4_solution.pdf>

2, <http://www.public.asu.edu/~ccolbou/src/355hw4s09sol.pdf>