CS350 – Turing Machines

Purpose: Turing machines comprise a computational model which is equivalent to Python programs. This means that everything that can be done in a Python program can be done on a Turing machine and everything that can be done on a Turing machine can be done with a Python program.

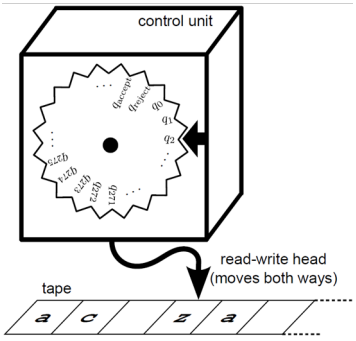
Due to their simplicity, we will sometimes find that it is easier to analyze the properties of computation with Turing machines rather than with a Python program. The purpose of this module is to explore the Turing machine model of computation.

Knowledge: This module will help you become familiar with the following content knowledge:

- How to use a Turing machine to accept a language
- How to use a Turing machine as a transducer in order to transform a string.
- How to use one model to simulate another different model of computation

Activity: With your group perform the following tasks and answer the questions. You will be reporting your answers back to the class in 60 minutes.

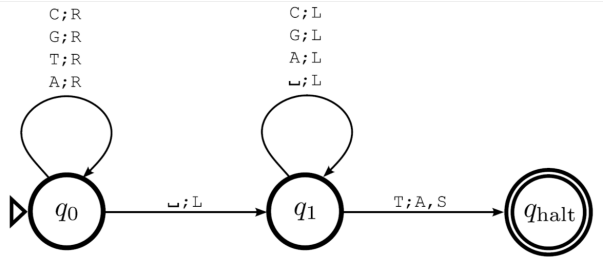
1. A Turing Machine has a tape and a read/write head which is at a particular location. It is also in a state at a particular time.



This Turing machine is in state q_2, with tape content "ac za", and the read/write head is at location 3.

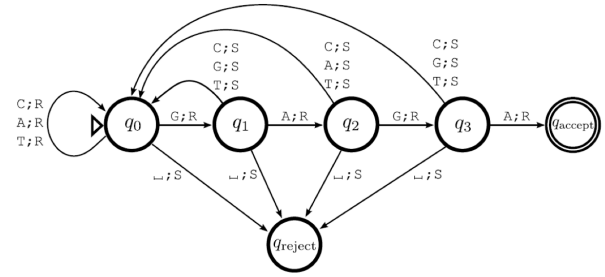
The Turing machine has an alphabet Σ which is a finite set of symbols, including a blank symbol.
The Turing machine has a set of states Q which includes a start state q_0, and may have one or more halting states.
The Turing machine has a transition function $\delta(q, x) : \delta(q, x) = (q', x', d')$ which says that if the machine is in state q and reading an x on the tape, then the machine moves into state q', writes x' at the head position, and moves the head in direction d' where the directions can be right,left, (and maybe stay in the same location).

Turing machines are easier to understand via *state diagrams*. The circles in the diagram represent states and the arrows represent transitions. For example, an arrow from state q_1 in the following diagram says that if the tape symbol at the head location is a T then write an A, stay in the same location and enter the halting state.



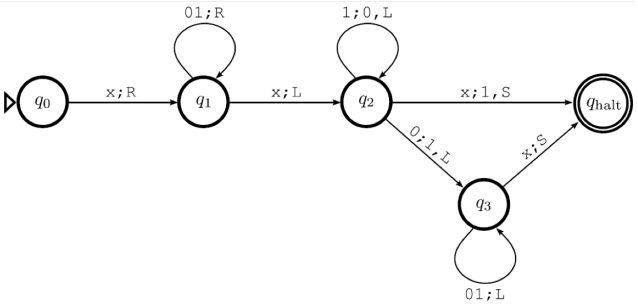
What does this TM do with the initial tape: "CTCGTA "?

2. Turing machines are often used as an acceptor to determine whether the contents on the tape is in a language or not. In this case, the state after scanning the tape contents will indicate that the string is accepted or rejected. (Accepting states are drawn with a double circle and there may be more than one accepting state).



What language does this Turing machine accept?

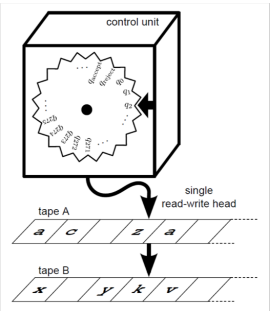
3. A Turing machine might instead be a transducer which modifies the contents of the tape.



What does this TM do with the tape "x110x"?
What does this TM do with the tape "x1101x"?
What does this TM do in general?

4. Why is the following statement FALSE?: If the input string is finite, then at some point, the Turing machine has to be able to finish reading it. Therefore, infinite looping can only happen when the input takes up the whole tape (which in infinitely long).

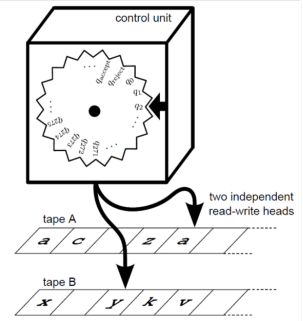
5. There are quite a few variations in the texts on how Turing machines operate but it turns out that all of them are equivalent. That means that one version of a Turing machine can be used to simulate the workings of another version. Consider a version of a Turing machine that has two tapes instead of one, but only a single read/write head.



M	tape A:	a	b	c	␣	d
	tape B:	u	␣	v	␣	w
M'	tape:	au	b␣	cv	␣␣	dw

Write a sentence (or two) explaining how a 2-tape TM can be simulated by a one tape TM.

Now consider a Turing machine that has two tapes, each with an independently move head.



M	tape A:	a	b	c	␣	**d**
	tape B:	u	␣	**v**	␣	w
M'	tape A_1:	**a**	b	c	␣	d
	tape A_2:	**␣**	␣	␣	␣	x
	tape B_1:	**u**	␣	v	␣	w
	tape B_2:	**␣**	␣	x	␣	␣

Write a sentence (or two) explaining how a 2-tape 2-head TM can be simulated by a multi-tape one head TM.

There are a lot of other simulation possibilities that include:

(a) A TM can simulate a python program running on a real computer

(b) A python program can simulate a TM

(c) Any known computer can simulate any other (so are called Turing equivalent).

Complete the following assignments for grading. Each should be done individually but you may consult with a classmate to discuss strategies. You will need to use JFLAP to construct the Turing Machines.

Assignment 1:
Complete exercise 5.1 on p99 of the text.

Criteria for Success: You have a JFLAP file for a Turing Machine with one tape that swaps the characters "C" and "G" and leaves all other characters unchanged. Make sure you test your Turing machine on multiple inputs so you are confident that it will work for all strings.

Assignment 2:
Create an acceptor Turing machine with the following properties: the machine accepts strings containing at least three G's and at most two T's and any number of A's and C's; all other strings are rejected.

Criteria for Success: You have a JFLAP file for a Turing Machine with one tape that accepts the given language. Use the TM states to keep track of how many G's and T's have been seen rather than trying to create a counter.
It is important to make sure in your Turing machine preferences that you accept by final state only – so uncheck the accept by halting in the preferences window.

Assignment 3:
Complete exercise 5.7 on p100 of the text.

Criteria for Success: You have a JFLAP file for a transducer Turing Machine which modifies the original input string so that after the TM is finished executing all the G's have been deleted. You do not want any spaces in the output of the TM, so the string "TTGAGA" would be transformed into "TTAA". A two-tape machine will be useful for this task.

Assignment 4:
Complete exercise 5.12 on p100 of the text.

Criteria for Success: You need a clear explanation on how a TM with an alphabet of four characters can simulate a TM with an alphabet of five characters. Notice that we are assuming that the input and output only use the four characters but the computation on the inputs may use all five characters.
Hint: You need to figure out how the characters in the 5-symbol alphabet could be "coded-up" with a 4-symbol alphabet.

Submit your JFLAP files and written answers in Canvas for grading.