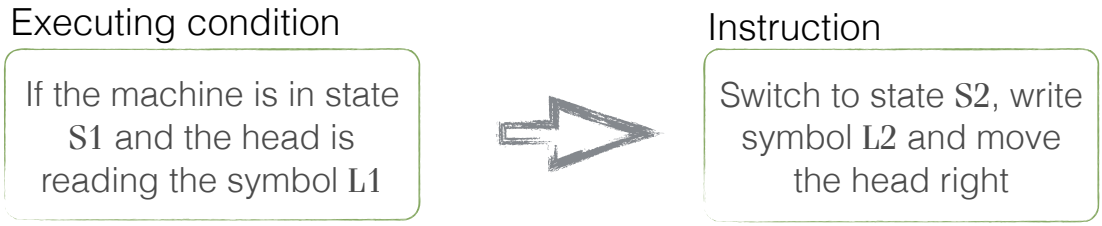
TURING MACHINEs With more than one tape

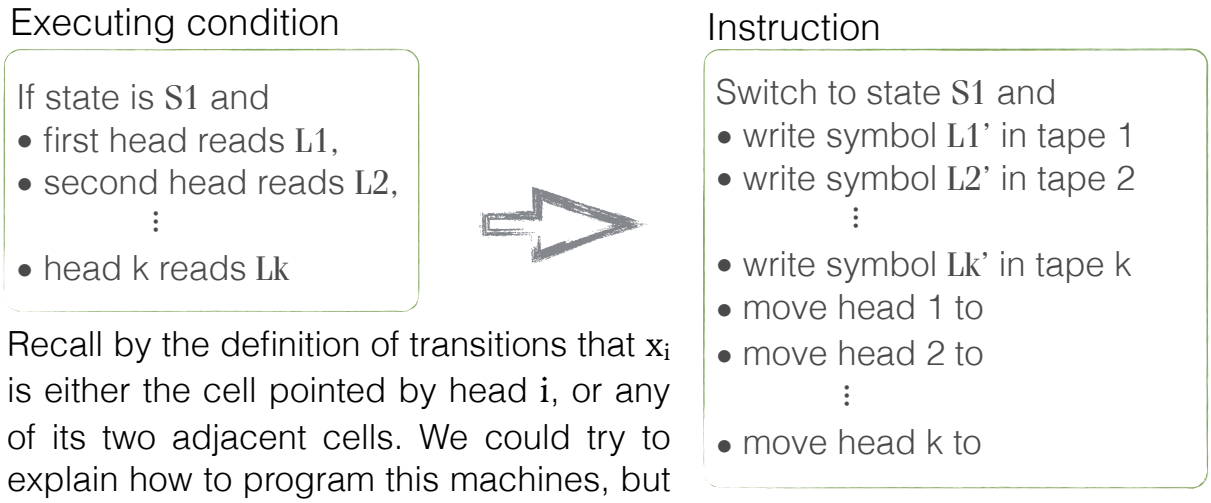
This section requires you to know what a TURING MACHINE is and how to program it. If you don't, feel free to visit the introduction and the tutorial.

Turing machines with multiple tapes are exactly what their name says. Probably if you load a two-tape example you'll understand it by yourself, but this section might save you some time. The only difference between single and multi tape machines is in the transitions. Recall that transitions in single taped machines are of the form



The executing condition mentions that the machine is reading just one symbol, which reflects the fact that this corresponds to a machine with a single tape. The same happens with the instruction as it writes one symbol.

In a Turing machine with k tapes, the condition of transitions depend on the state of the machine and on the k symbols being read. Moreover, the instruction changes the state, writes k symbols, and move the head of the k tapes. This means that transitions have the next form:



Recall by the definition of transitions that x_i is either the cell pointed by head i, or any of its two adjacent cells. We could try to explain how to program this machines, but it's probably easier to see an example.

Figure 1 shows the coding of a Turing machine with two tapes. You can notice that transitions' conditions now have three elements, one state and two symbols to read. Furthermore, the instructions have five elements: one state to adopt, two symbols to write and two 'directions' to move the first and second head, respectively. Figure 2 shows how this machine would look like after it is compiled and the input '0010100' is loaded. Notice that the input is loaded into the first tape. In Turing machines with multiple tapes, the input is loaded in the first tape and every other tape will always start empty (with every cell blank).

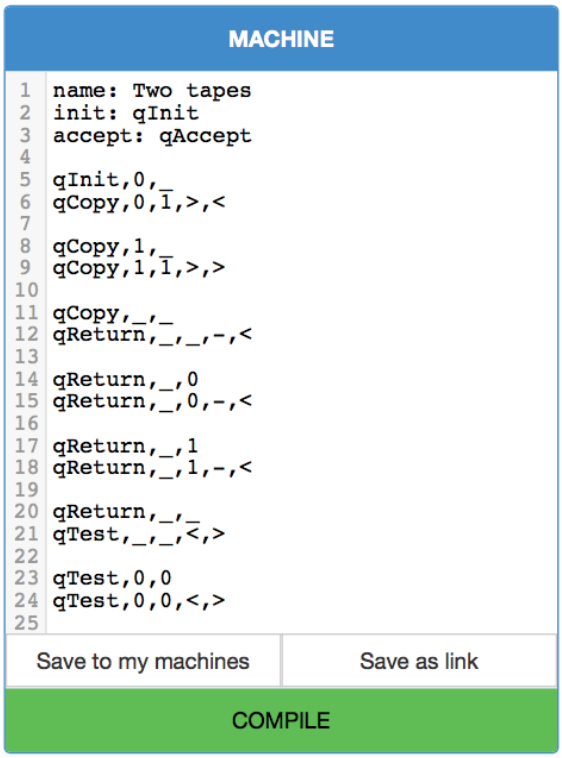


Figure 1

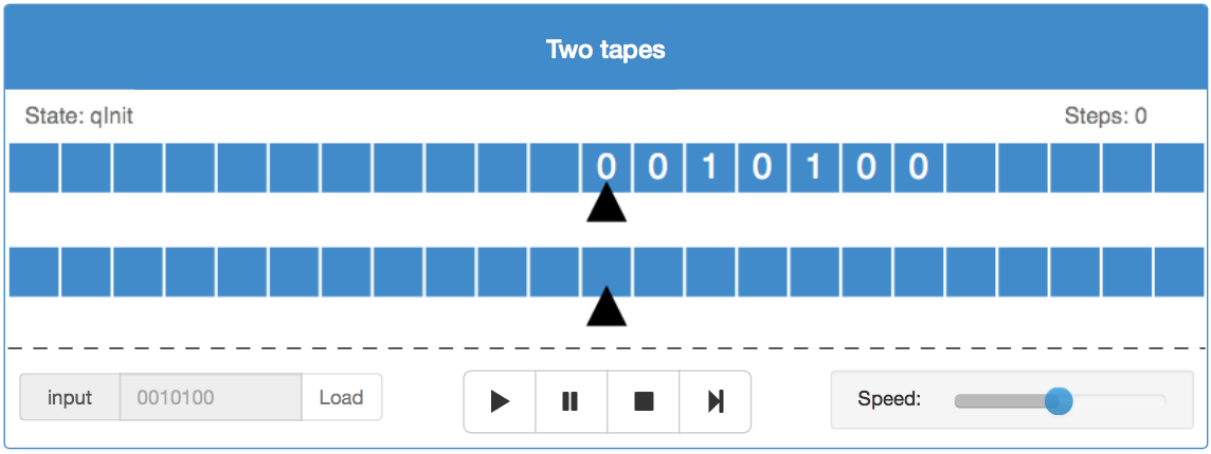
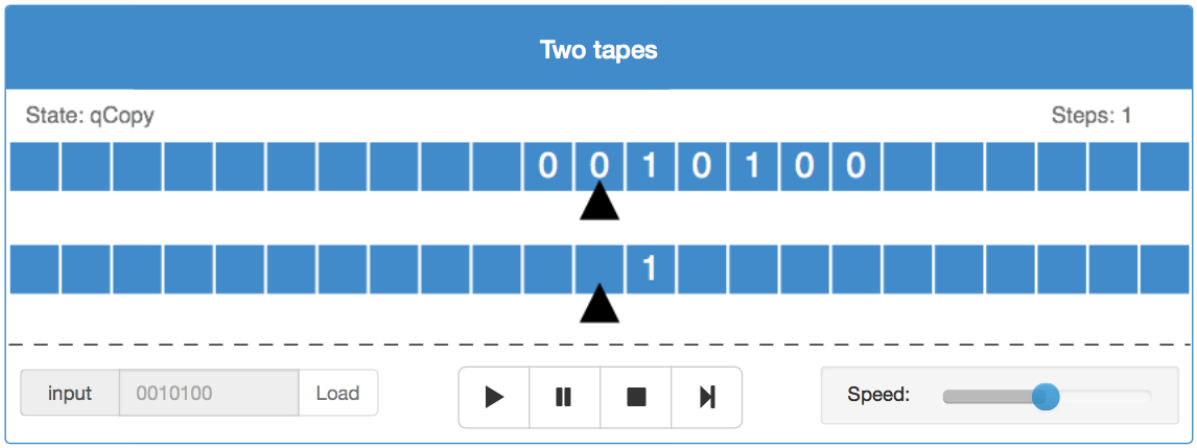


Figure 2

At the beginning of the run, the machine's state is qInit and the symbols being read are a zero and a blank. Thus, the condition that will trigger an instruction is 'qInit,0,_'. This is the condition of the first transition, which's instruction will be executed. Figure 3 shows how the machine looks like after this transition is applied. Notice that the symbols and the head movement directions in the transitions correspond to the tapes in a top-down fashion (i.e. the first symbol corresponds to the topmost tape).



We've shown just one transition of a Turing machine with multiple tapes, but considering you already know about single taped machines this should be enough to deduce everything you need to know. Maybe you are wondering why the machine has just one state instead of one per tape. This is because using multiple states just makes the model more complex, it does not provide more computational power and doesn't simplify the coding.

I sincerely hope that after reading the first four sections you are capable of programming your own Turing machines and modifying other's. If you are interested in a more theoretical point of view about Turing machines, please feel free to read the theoretical sections, starting with the introduction to formal languages.