1. (See 1.png)

2. (See 2.png)

	0	1
q_0	q_0	q_1
q_1	q_2	q_1
q_2	q_0	q_3
q_3	q_2	q_4
q_4	q_4	q_4

4. 0*10*

↑

Does 0* allow for no 0's, too?

edit: yes! looks like * indicates <u>0 or more</u> occurrences.

3. A machine that could accept strings over a larger alphabet would be more powerful than a machine that could only accept strings over the alphabet {0,1} in some cases, but be about equally as powerful in others. Assuming letters can be represented as bits in binary, and vice versa (ie. let a=0, b=1, etc.), any letter in binary could be represented by a string of 8 1's and 0's, while any string of numbers could be represented by a string of a's and b's of the same length. There is an increase in the number of bits being processed in the letter-to-numbers conversion, while there is no change in the numbers-to-letters conversion. Therefore, I think that except in cases where the input is very simple and there is only a single (or around a single) bit to process, machines that accept strings over larger alphabets are more powerful than machines that only accept {0,1}.