Computing Functions in Turing Machines

Computing functions in TM is somewhat different from accepting a language by a TM. In accepting a string of the language the TM has to go to Halt state. Thus going to a halt state is same as saying "yes", i.e, the string is accepted. If it does not go to a halt state, it means "no", i.e, the string is not accepted. It does not matter what is left on the tape at the time the TM halts. In computing a function, however, you have to come up with a mechanism to present the result (output) of the function. The output for a TM is left on the same tape replacing what ever was there before. Representing the input is also different from a TM accepting a string of a language, because you may have more than one values that you need for input. For example, for the function f(x,y)= x+y, you have to have a way of representing the two numbers x and y on the input Tape before the TM starts and a convention for leaving the result (i.e., x+y) on the Tape when the TM halts after the computation. Following is a convention for handling the problem. I will explain it with an example for computing f(x,y)=x+y

We will assume that the numbers x and y are unsigned integers. We will represent an integer by the number of 1 bits corresponding to the value of the integer. Thus, the integer zero is represented by zero number of 1’s (i.e., no 1’s), the integer 1 by one 1 bit, 2 by two 1 bits (i.e 11), 3 by three 1 bits (i.e 111) and so on. You could have used a different conversion where integer 0 will be represented by one 1, integer 1 by two 1’s, integer 2 by three 1’s and so on. For the input to the TM, we will place the two integers x and y on the tape separated by a Δ. Thus, the separating Δ is needed by the TM to distinguish the first number from the second. If we want to add x and y where x=2 and y=3, the input tape will be as follows (note that the output will be left on the same tape replacing this input):

Δ11Δ111ΔΔ- - -

After the TM has finished computation on this input, the result 5 will be left on the tape left justified as follows. Also note that we did not save the input on the tape.

Δ11111ΔΔΔ- - -

Actual computation by this TM is rather very simple. Only thing the
TM has to do is to convert the input above into an output which leaves five 1’s on the tape. Essentially, this means that the TM has to do a shift of all the bits in y one position to the left on the Tape. Simpler even, the TM will replace the Δ that separates x and y by a bit 1 and replace the last bit of y (i.e., right most 1 of the integer y) by a Δ. This is same as the left shift because all bits in y are 1.

To compute the function \( f(x) = x + 2 \), for example, the TM has to replace the two Δs right of the input x by two 1’s and then halt. Note that the function here has only one input parameter x. The integer 2 that is to be added to x is not a part of the input.

We should be able to design Turing Machines if the numbers are stored as binary integers, instead. For example the integer 2 may be stored as the binary 10. But the Turing Machine may be more complex to compute a function using this representation. But the point is that TM can be designed to handle other types of input format.