Problem 1

The computation of the addition proceeds in 4 main steps: an initialize step, a main addition step, a secondary addition step, and, for lack of a better term, a finish step. Each step will be described separately.

The Turing machine computes each bit in turn and replaces the left number one bit at a time with the sum. See Figure 1.

The initialize step is very simple, it consists solely of state q0. Since the head of the turing machine starts at the left of the input, state 0 moves the head to the right until it reads the # at which point it moves left to the least significant bit of the first number.

The main addition step happens while there are still bits left unprocessed in the first number. States 1 through 12 handle the main addition step. States 1 and 2 correspond to a carry bit of 0 and 1, respectively. The TM reads the bit under the head, marks it with an x, adds it with the carry bit, scans to the last nonblank cell, and adds that bit while blanking it out. At this point, the TM is in one of the states 6, 7, 8, or 12, corresponding to \((c, s) = (0, 0), (0, 1), (1, 0), \) or \((1, 1)\) where \(c\) is the carry bit and \(s\) is the sum bit.

The last part of the main addition step is to scan left back to the \(x\), write the sum bit to the tape in place of the \(x\), and go to state 1 or 10, depending on the carry bit.

Note that if all of the bits of the second number had already been blanked out, the bit is treated as 0.

The secondary addition step involves only the remaining bits of the second number and the carry that resulted from the last step of the main addition step. States 13–19 handle this step. The computation is the same as the main addition step except that there is only one number to add with the carry bit.

For the finish step, all of the bits have been added and the head is at the #. It overwrites the # with a blank, scans to the left, and if needed, writes the carry bit.

Problem 2

To show that the problem is undecidable, we need to formulate it as a language. Let

\[ L = \{ \langle Q, x, B \rangle \mid Q \text{ is a computer program containing the variable } x \text{ and } \exists w \in \Sigma^* \text{ such that when } Q \text{ is run on } w, x \text{ is assigned a value larger than } B \}. \]

Now, we reduce \( \text{HALT}_{\text{TM}} \leq L \). Let \( P \) decide \( L \) and build a Turing machine \( H \) to decide \( \text{HALT}_{\text{TM}} \) as follows, \( H = \text{“On input } \langle M, w \rangle, \text{ do the following:} \)

1. Let \( x \) be a variable not used in \( M \).
2. Build program \( Q = \text{‘On any input,} \)
   
   \[ \begin{align*}
   &1. \quad x \leftarrow 0. \\
   &2. \quad \text{Ignore the input and run } M(w). \\
   &3. \quad x \leftarrow 1.
   \end{align*} \]

3. Run \( P(\langle Q, x, 0 \rangle) \) and output what \( P \) outputs.”

If \( M(w) \) halts, then \( x \) will be set to 1 which is greater than the bound \( B = 0 \) so \( P \) will output “yes.” Otherwise, \( x \) will never be set to 1 and thus \( P \) will output “no.” Since \( \text{HALT}_{\text{TM}} \) is undecidable, \( L \) is undecidable.
Figure 1: Turing machine for problem 1.