James Yao

Writing assignment – Programming assignment 7

Completing programming assignment seven involved creating a Turing machine that subtracted two numbers. The input string was formatted as X#Y where X and Y are base three numbers (digits were zero, one, or two). The output string must contain X – Y. When the Turing machine accepts, the head must be located at the start of the solution.

To implement this Turing machine, I first inserted a '$' marker at the start of the tape. I checked if both X and Y were zero; if so, position the head at the last symbol in our input (it should be a '0'). Next, I checked if Y was zero; if so, replace Y and '#' with blank characters and position the head at the beginning of X past all leading zeros. Finally, I checked if X was zero; if so, move to the beginning of Y, move past all leading zeros, put a '–' in front of the rest of Y, and position the head on the negative symbol. Our edge cases of X = 0, Y = 0, or X = Y = 0 are now functional.

To figure out if X > Y or Y < X, I first removed all leading '0's in X and Y. Zeros in X became '$'s and zeros in Y became '#'s. Next, for each symbol in X, I marked it and I marked a corresponding symbol in Y. The marking process was '0' became 'a', '1' became 'b', and '2' became 'c'. If I ran out of symbols in Y first, X > Y and we do not need to adjust our input. A '#' is inserted at the end of our input to indicate the location of our future solution. If I ran out of symbols in X first, I check if there is another symbol in Y. If so, Y > X and I insert a '#' at the end of the input. I then copy all symbols in X to the location after this '#' (the original X and '#' is overwritten with '$'s). Now our input is Y#X where Y > X. Another '#' is added to the end of the tape and following it, a '–' is added as well to indicate our

solution will be negative. In the case where another symbol in Y was not found and the length of X equals the length of Y, I find the first symbol that is different between X and Y starting from the left. If none are found, insert a zero at the end of the tape and accept. If a difference is found, determine which is larger (‘c’ > ‘b’, ‘b’ > ‘a’, etc.). If X is larger, add ‘#’ to the end of the tape. If Y is larger, do the same as we did above when length of Y was larger (copy X over and add ‘#’ and ‘–’).

Now all that is left is to do the subtraction between two numbers where the first is larger. I will call the larger (left number) X and the smaller Y. Starting from the rightmost digit in X, I subtract it with the rightmost digit in Y. Two minus one is zero, one minus zero is one, zero minus one is equal to three minus one (add three to the digit in X if it is smaller than the one in Y), etc. If the digit in X if it is smaller than the one in Y we also need to subtract one from the digit immediately left of the digit in X we just used. If that digit is zero, repeat for the next left digit. We will always eventually find a non-zero digit because X > Y. The result of each computation is stored at the end of the tape. Finally, the ‘–’ is copied to the end of the tape and the results are copied in reverse to the end of the tape. We do this because we computed the differences from least significant digit to most. If there are leading zeros in the flipped solution and the solution is negative, replace them with ‘–’s. Finally, position the head at the last ‘–’ or at the first nonzero digit if the solution is positive.

Testing the program was simple, but I did not do so with every possible configuration of needing a different carry over digits (‘0’ – ‘2’ was programmed to give the wrong result and I did not notice until later). I manually did the subtraction to make sure my Turing machine’s answer was correct.