Unary to Decimal Converter Turing Machine

Preprint · October 2023

DOI: 10.13140/RG.2.2.15740.14725

1 author:

Mohammad Karbalaee
Shahid Beheshti University
20 PUBLICATIONS **0** CITATIONS
SEE PROFILE

Shahid Beheshti University

Computation Theory

Dr. Dastgheib

Mohammad Karbalaee

5th Assignment

October 29, 2023

1 Problem statement

In essence, we aim to devise a Turing machine capable of determining the length of a given input string, which is invariably composed of consecutive "1"s. For instance, if we provide the input "111," the machine should yield the output "3." Consequently, the machine's output is a positive integer.

This Turing machine will extract the count directly from the tape during its operation. As a result, upon completion, the tape should contain nothing other than the counted numerical value.

2 Solutions

Initially, I conceived an approach to address this issue, only to later recognize its impracticality. It became evident that this approach would necessitate an excessively vast number of states, rendering the machine unmanageable and unacceptable. Nonetheless, I believe it's worth noting this initial concept as it might yield valuable insights in the future.

Subsequently, I devised an alternative approach, one that seemed promising for solving the problem. Regrettably, I encountered substantial technical complexities and encountered significant hurdles, preventing me from reaching a definitive conclusion.

2.1 The Naive Approach

2.1.1 Overview

The Turing machine I devised for this problem functions effectively for strings with a size of three or less. However, I encountered a critical limitation as I sought to apply it to larger strings.

During the machine's construction, I observed a recurring pattern in which I needed to continually add states to accommodate larger strings. This phenomenon resembled entering an infinite loop of state expansion.

This issue arose due to the nature of the problem: for each increment of the counter, a comparison between the previous count and the new count is required. Consequently, we find ourselves creating state branches from one count to the next, which ultimately renders the solution impractical for processing larger numbers.

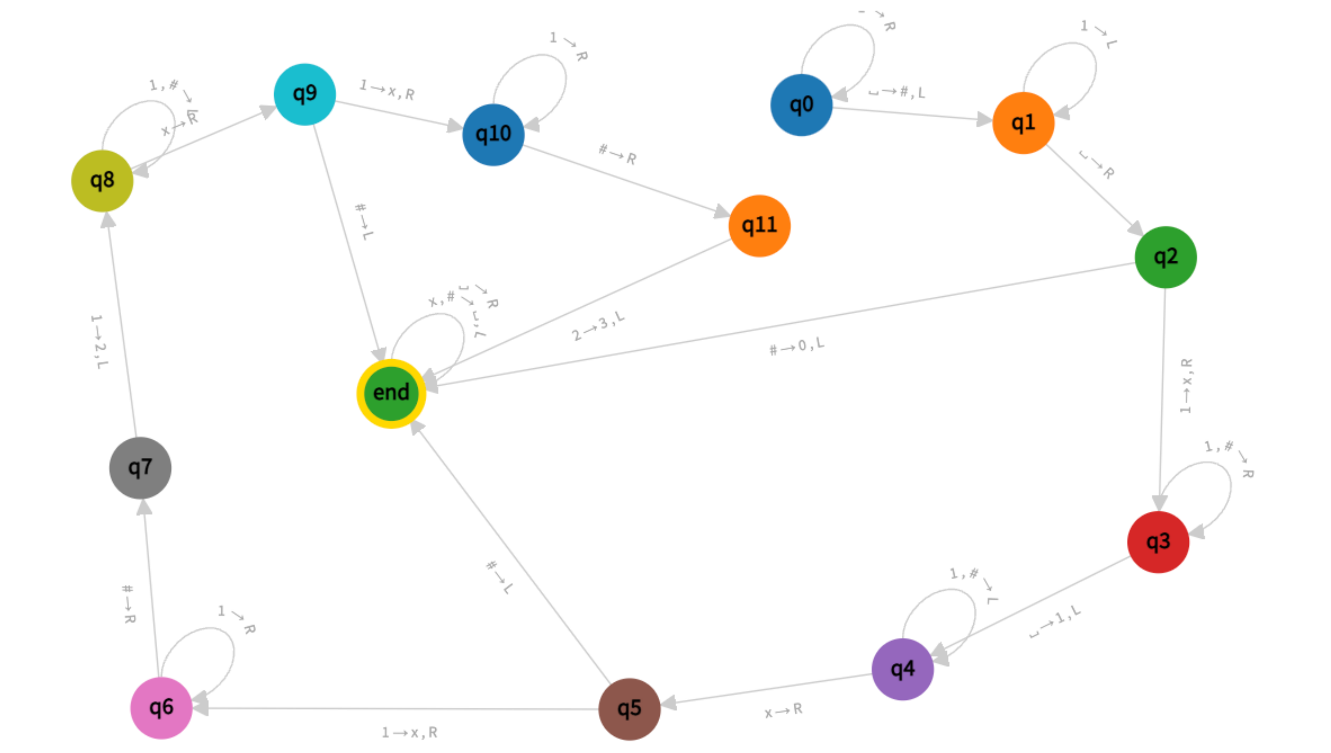

Figure 1: The Turing machine that counts strings of size 3 or less

2.1.2 How it works?

The steps of this machine can be defined as below:

- Finding the end of the input string and marking it with a ”#” symbol.
- Reading ”1”s from left to right, one by one, and replacing each of them with ”x.”
- Incrementing the counter, which is located right after the end of the string marked by ”#.”
- When there are no ”1”s left, the counter displays the correct count.
- Finally, all other elements except the count value are removed.

2.1.3 Attached Material

Alongside this document, I have included a brief video demonstrating the operation of the Turing machine I designed. Additionally, I have shared the code I created to visually represent the machine's functionality. Below, you will find the code I used to visualize this Turing machine:

```
input: '111'
blank: ' '
start state: q0
table:
  q0:
    1: R
    ' ': {write: "#" , L: q1}
  q1:
    1: L
    ' ': {R: q2}
# now we know the end of the string
  q2:
    1: {write: "x", R: q3}
    "#": {write: "0", L: end}
  q3:
    [1, "#"]: R
    ' ': {write: "1", L: q4}
  q4:
    [1, "#"]: L
    "x": {R: q5}
  q5:
    1: {write: "x", R: q6}
    "#": {L: end}
# this is where processing strings with length of 1 finishes
  q6:
    1: R
    "#": {R: q7}
  q7:
    1: {write: "2", L: q8}
  q8:
    [1, "#"]: L
    "x": {R: q9}
  q9:
    1: {write: "x", R: q10}
    "#": {L: end}
# this is where processing strings with length of 2 finishes
  q10:
    1: R
    "#": {R: q11}
  q11:
    "2": {write: "3", L: end}
# this is where processing strings with length of 3 finishes
  end:
    ["x" ,"#"]: {write: ' ', L: end}
    ' ': R
```

2.2 The Better Approach

2.2.1 Overview

Taking an alternative perspective, it becomes apparent that a string consisting solely of ”1”s represents a unary number. By transforming this unary number into our desired format, decimal, we can effectively address the problem.

In my initial attempt, I embarked on creating a Turing machine for the conversion of unary numbers into decimal numbers, but this endeavor did not yield success. Subsequently, I considered that the conversion from unary to binary numbers is a more tractable task. Once in binary format, I envisioned constructing a separate Turing machine for converting binary to decimal. My plan involved the integration of these two machines to arrive at the ultimate solution.

Below you can find both of these machines.

2.2.2 Unary-binary converter

Below you can see a diagram of this machine. Also you can find the code and sample video about this machine.

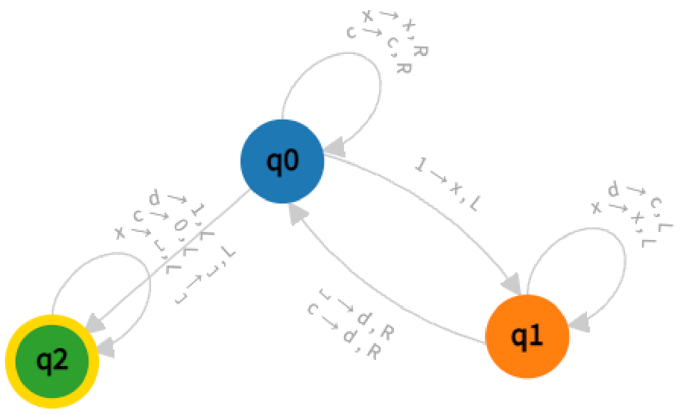

Figure 2: Unary-binary number converter

```
input: '111'
blank: ' '
start state: q0
table:
  q0:
    "1": {write: "x", L: q1}
    "c": {write: "c", R}
    "x": {write: "x", R}
    ' ': {write: ' ', L: q2}
  q1:
    "x": {write: "x", L}
    "d": {write: "c", L}
    ' ': {write: "d", R: q0}
    "c": {write: "d", R: q0}
  q2:
    "x": {write: ' ', L}
    "c": {write: "0", L}
    "d": {write: "1", L}
```

Below is a Turing machine that converts binary numbers to decimal ones.

By initiating this Turing machine after converting unary to binary numbers, we get a new machine that can completely fit our needs.

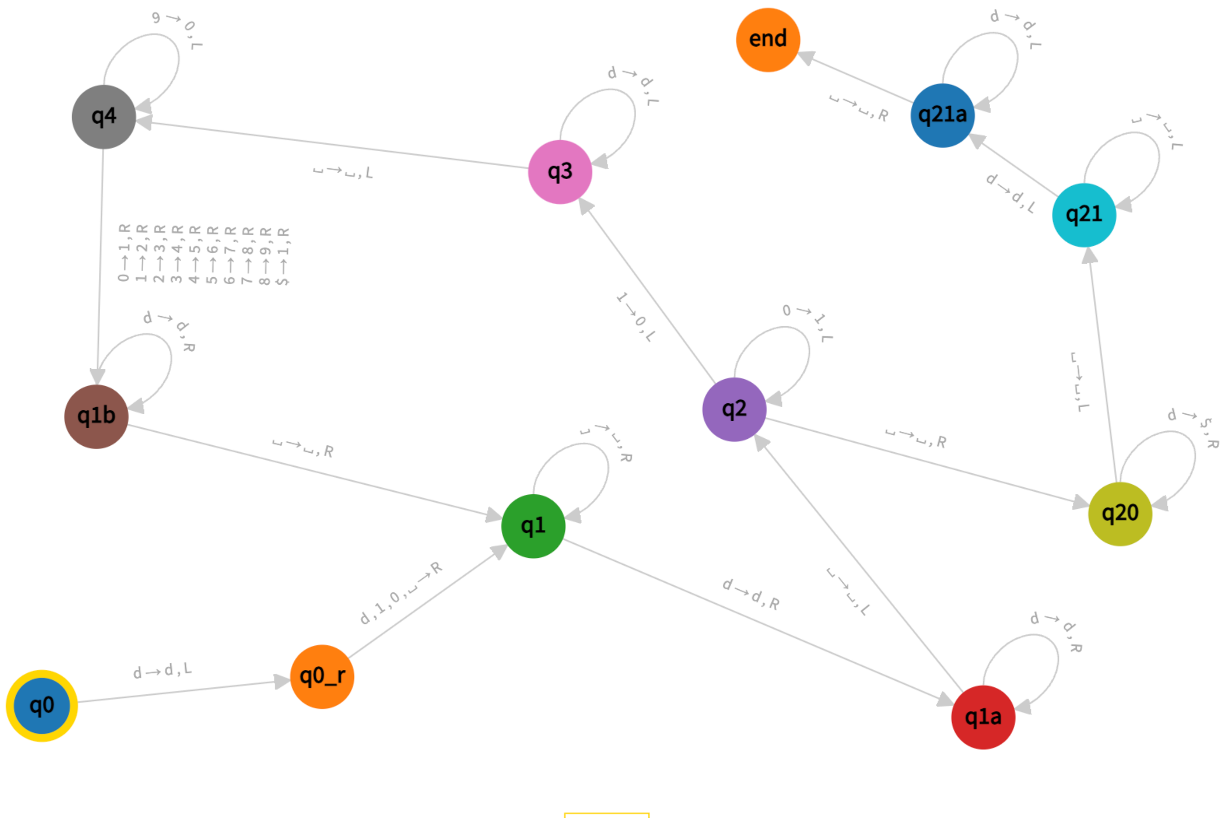

Figure 3: Binary to Decimal converter

Below is the code for this simulation.

```
# d stands for 1
input: 'd0d'
blank: ' '
start state: q0
table:
  q0:
    'd': {write: 'd', L: q0_r}
  q0_r:
    ['d', "1", "0", " "]: {R: q1}
  q1:
    " ": {write: " ", R: q1}
    'd': {write: 'd', R: q1a}
  q1a:
    'd': {write: 'd', R: q1a}
    " ": {write: " ", L: q2}
  q2:
    "1": {write: "0", L: q3}
    "0": {write: "1", L: q2}
    " ": {write: " ", R: q20}
  q1b:
    " ": {write: " ", R: q1}
    'd': {write: 'd', R: q1b}
  q3:
    'd': {write: 'd', L: q3}
    " ": {write: " ", L: q4}
  q4:
    "0": {write: "1", R: q1b}
    "1": {write: "2", R: q1b}
    "2": {write: "3", R: q1b}
    "3": {write: "4", R: q1b}
    "4": {write: "5", R: q1b}
    "5": {write: "6", R: q1b}
    "6": {write: "7", R: q1b}
    "7": {write: "8", R: q1b}
    "8": {write: "9", R: q1b}
    "9": {write: "0", L: q4}
    "$": {write: "1", R: q1b}
  q20:
    " ": {write: " ", L: q21}
    'd': {write: "$", R: q20}
  q21:
    " ": {write: " ", L: q21}
    'd': {write: 'd', L: q21a}
  q21a:
    'd': {write: 'd', L: q21a}
    " ": {write: " ", R: end}
  end:
```