Math 29: Computability Theory **Spring 2024**

PSET 2 — 04/12/2024

Prof. Miller ***Student: Amittai Siavava***

Problem 1.

Show that the Fibonacci function, where $f(0) = f(1) = 1$ and $f(n+2) = f(n+1) + f(n)$ is computable by building a register machine.

Idea

The register machine in Figure 1 works using bottom-up ***dynamic programming***. At the start, we set R_1 to 1. We then iterate n times, at each step setting R_1 to contain a new value equal to the previous values of R_1 and R_2 added together, and setting R_2 to previous value of R_1. After iterating n times (i.e. when deducting 1 from R_0 is impossible), we stop and R_1 contains the value of $f(n)$.

Note: for the register machine and the more-detailed algorithm below, I used R_{11} and R_{12} to store intermediate values for R_1 and R_2 respectively. This is to avoid overwriting the values of R_1 and R_2 while they are still being used in the current iteration.

Algorithm 1: Compute $f(0) = f(1) = 1$ and $f(n+2) = f(n+1) + f(n)$

start
$R_1 \leftarrow 1$
while $(\textit{status} := R_0 - 1) \neq e$ **do** ▷ `iterate` n `times`
  while $(\textit{status} := R_1 - 1) \neq e$ **do** ▷ `add` R_1 `to` R_{11} `and` R_{12}
    $R_{11} \leftarrow R_{11} + 1$
    $R_{12} \leftarrow R_{12} + 1$
  while $(\textit{status} := R_2 - 1) \neq e$ **do** ▷ `add` R_2 `to` R_{11}
    $R_{11} \leftarrow R_{11} + 1$
  while $(\textit{status} := R_{11} - 1) \neq e$ **do** ▷ `move value from` R_{11} `to` R_1
    $R_1 \leftarrow R_1 + 1$
  while $(\textit{status} := R_{12} - 1) \neq e$ **do** ▷ `move value from` R_{12} `to` R_2
    $R_2 \leftarrow R_2 + 1$
stop

Register Machine

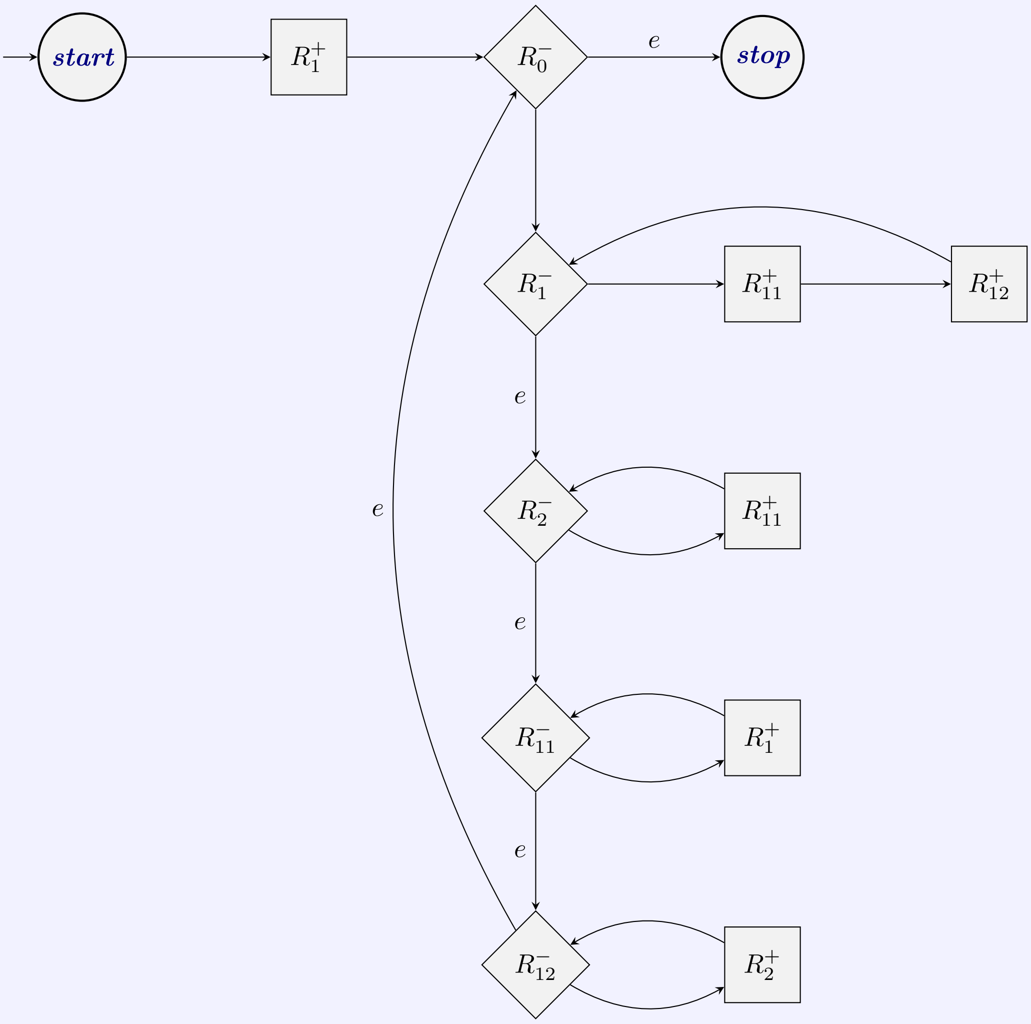


Figure 1: Compute $f : n \mapsto \begin{cases} 1 & \text{if } n \in \{0, 1\} \\ f(n-1) + f(n-2) & \text{otherwise.} \end{cases}$

Problem 2.

Show that the set of powers of 2 is computable by building a Turing machine.

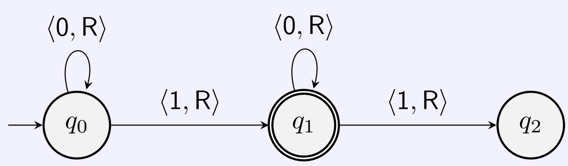
Idea

A number (in binary) is a power of 2 if and only if it has exactly one 1 bit. Therefore, we can construct a Turing machine as follows: that scans the input tape from left to right, counting the number of 1 bits it encounters.

1. The Turing Machine starts in state q_0.
2. While in q_0, it reads the tape at the current position.
   (a) If it reads a 0, it moves right and remains in q_0.
   (b) If it reads a 1, it moves right and transitions to q_1.
3. While in q_1, it reads the tape at the current position.
   (a) If it reads a 0, it moves right and remains in q_1.
   (b) If it reads a 1, it moves right and transitions to q_2.
   (c) If it reads a blank symbol, it halts in q_1.
   This is an accepting scenario since the Turing machine has encountered exactly one 1 *bit in the entire binary string.*
4. q_2 is a non-accepting state without any transitions. If in q_2, it does not matter what the Turing machine reads—the string is not a power of 2 since it already has more than one 1 bit. Thus, reading any symbol while in q_2 would cause it to halt and not accept the input as a power of 2.

Turing Machine

1. $\langle q_0, 0, \mathsf{R}, q_0\rangle$
2. $\langle q_0, 1, \mathsf{R}, q_1\rangle$
3. $\langle q_1, 0, \mathsf{R}, q_1\rangle$
4. $\langle q_1, 1, \mathsf{R}, q_2\rangle$



Problem 3.

Show that the set of multiples of 4 is computable by building a Turing machine.

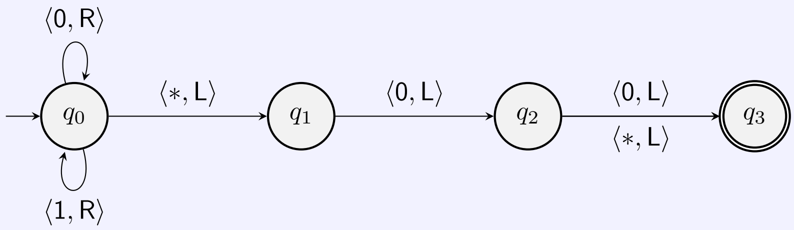
Idea

A number (in binary) is a multiple of 4 if either the number is 0 or the last two bits are 0. We check this by reading the input from left to right, until we reach the end of the input, then checking the last two bits on the tape.

1. The Turing machine starts in q_0. It reads the tape at the current position.
   (a) If it reads a 0, it moves right and remains in q_0.
   (b) If it reads a 1, it moves right and remains in q_0.
   (c) If it reads a blank ($*$), it moves left on the tape and transitions to q_1.
2. While in q_1, it reads the tape at the current position.
   (a) If it reads a 0, it moves left and transitions to q_2.
   (b) If it reads a 1, or a blank symbol ($*$), it halts in q_1 and does not accept.
3. While in q_2, it reads the tape at the current position.
   (a) If it reads a 0 or a blank ($*$), it moves right and transitions to q_3.
   NOTE: we include the empty symbol $*$ here to account for the number 0.
   (b) If it reads a 1, it halts in q_2 and does not accept.
4. q_3 is an accepting state. If the Turing machine reaches q_3, whatever it reads next halts the machine and accepts the input as a multiple of 4.

Turing Machine

1. $\langle q_0, 0, \mathsf{R}, q_0 \rangle$
2. $\langle q_0, 1, \mathsf{R}, q_0 \rangle$
3. $\langle q_0, *, \mathsf{L}, q_1 \rangle$
4. $\langle q_1, 0, \mathsf{L}, q_2 \rangle$
5. $\langle q_2, 0, \mathsf{L}, q_3 \rangle$
6. $\langle q_2, *, \mathsf{L}, q_3 \rangle$



Problem 4.

Give a numerical code for the Division register machine provided in Monday's class according to the coding scheme established Friday. *(Do NOT multiply it out into decimal!)*

Register Machine

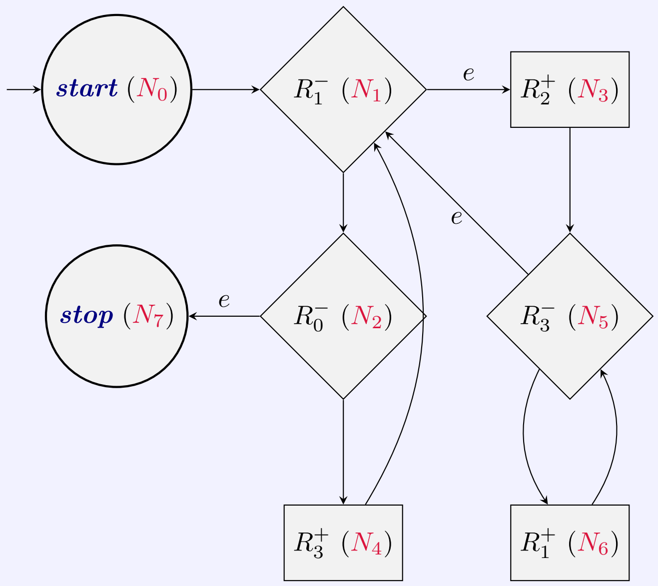


Figure 2: Register machine M that computes division.

The numbers in parentheses are the references for each node.

Code

$$\#N_1 \mapsto 2 \times 3^1 \times 5^2 \times 7^3$$

$$\#N_2 \mapsto 2 \times 3^0 \times 5^4 \times 7^7$$

$$\#N_3 \mapsto 3^2 \times 5^5$$

$$\#N_4 \mapsto 3^3 \times 5^1$$

$$\#N_5 \mapsto 2 \times 3^3 \times 5^6 \times 7^1$$

$$\#N_6 \mapsto 3^1 \times 5^5$$

$$\begin{aligned}\#M &= \prod_{i=1}^{n} p_i^{\#N_i} \\ &= 3^{2\times 3^1 \times 5^2 \times 7^3} \times 5^{2\times 5^4 \times 7^7} \times 7^{3^2 \times 5^5} \times 11^{3^3 \times 5^1} \times 13^{2\times 3^3 \times 5^6 \times 7^1} \times 17^{3^1 \times 5^5}\end{aligned}$$

Problem 5.

Describe informally what process you would use to determine if the register machine coded by n contains a subtraction node. You may assume that the n you are given is a valid code for a register machine. *You do not need to provide a machine which runs your process.*

The encoding of nodes is done as follows:

(i) If N_i is an addition node R_j^+ with output node N_k, then $\#N_i = 3^j \times 5^k$.

(ii) If N_i is a subtraction node R_j^- with output node N_k and empty output node N_l, then $\#N_i = 2 \times 3^j \times 5^k \times 7^l$.

(iii) The encoding of M is then computed as $\#M = \prod_{i=1}^{n} p_i^{\#N_i}$.

In particular, a node N_i is a subtraction node *if and only if* its encoding $\#N_i$ is divisible by 2. Given an encoding $\#M$ of a register machine M, we can determine if M contains a subtraction node as follows:

1. Compute the prime factorization of $\#M$.

2. Iterate through each prime factor p_i and check if its exponent, equivalent to $\#N_i$, is divisible by 2. If it is, then node N_i is a subtraction node.

3. If none of the prime factors have an exponent divisible by 2, then the register machine M does not contain a subtraction node.