Turing Machines for Computation

Prof. Shrisha Rao
IIIT Bangalore

srao@iiitb.ac.in

2025-09-11

Exercises

(1) Come up with a TM that computes the bitwise complement of any binary string given to it.

(2) Come up with a TM that halts if the binary input given to it is divisible by 4, and does not halt otherwise.

(3) Come up with a TM for the language $(0+1)^* \setminus \{0^n 1^n \mid n \geq 1\}$.

(4) Prove that the function $\lfloor \frac{x}{2} \rfloor$ is Turing computable, by giving a TM that computes it.

As before, come up with the TM logic and pseudocode first, and then only draw the diagram. Show the working of the TM step by step.

Solution to #1

Example of TM Execution: input = 0110; initial tape configuration

$$\cdots \Delta \overset{\downarrow}{0} 1\,1\,0\,\Delta \cdots.$$

TM move logic:

1. Start from the leftmost bit.
2. If the current bit is 0, change it to 1.
3. If the current bit is 1, change it to 0.
4. Move to the right to the next bit.
5. Repeat steps 2–4 until the blank symbol Δ is encountered, signaling the end of the string.
6. Accept and halt.

Solution to #1

Example of TM Execution: input = 0110; initial tape configuration

$$\cdots \Delta \overset{\downarrow}{0} 1 1 0 \Delta \cdots.$$

1. In state q_0, head reads 0, writes 1, moves right. Tape after this step reads $\cdots \Delta 1 \overset{\downarrow}{1} 1 0 \Delta \cdots$.
2. In state q_0, head reads 1, writes 0, moves right. Tape after this step reads $\cdots \Delta 1 0 \overset{\downarrow}{1} 0 \Delta \cdots$.
3. In state q_0, head reads 1, writes 0, moves right. Tape after this step reads $\cdots \Delta 1 0 0 \overset{\downarrow}{0} \Delta \cdots$.
4. In state q_0, head reads 0, writes 1, moves right. Tape after this step reads $\cdots \Delta 1 0 0 1 \overset{\downarrow}{\Delta} \cdots$.
5. In state q_0, head reads Δ, writes Δ, stays in place, accepts.

Solution to #4

1. Point to note: in unary alphabet where x is given by $x+1$ 1s, the answer $\lfloor \frac{x}{2} \rfloor$ will have $\lfloor \frac{x}{2} \rfloor + 1$ 1s.
2. The simplest approach is to have the head overwrite the first 1 by a new symbol (e.g., A) and the last 1 by B, successively. As this goes on, there are two cases.
3. If there are an even number of 1s (meaning an odd input x), there will be equal numbers of As and Bs. In this case, delete all the Bs, convert all the As to 1s, and we are done.
4. If there are an odd number of 1s (meaning an even x), there will be a single 1 in the middle surrounded by equal numbers of As and Bs. In that case, delete all the Bs, change all the As to 1s, and we are done.

Solution to #4–cont'd

- Start with the head on the leftmost cell of the input (a block of 1s).
- **Repeat:**
  - Scan right from the left end to find the first unmarked 1. If none is found (you hit Δ), go to Cleanup.
  - Replace that first unmarked 1 by A.
  - Continue scanning right to find the last unmarked 1 (i.e. the rightmost 1 you encounter).
    - If you find a second unmarked 1, replace it by B, then return the head to the leftmost cell and repeat the loop.
    - If you reach a blank Δ before finding a second unmarked 1, there was a single leftover 1 which you have already turned into A; go to Cleanup.
- **Cleanup:** Move to the leftmost nonblank cell and scan right, performing:
  - Replace every A by 1.
  - Replace every B by Δ (erase it).
- Halt and accept. The tape now begins with $\lfloor x/2 \rfloor + 1$ ones which, under our encoding, is the unary code for $\lfloor x/2 \rfloor$.

Solution to #4–cont'd

TM specification:

- Input alphabet: $\Sigma = \{1\}$.
- Tape alphabet: $\Gamma = \{1, A, B, \Delta\}$ (where Δ is blank).
- States: ?
- Start state: q_0. Accept state: q_{acc}.

Give the bubble diagram for this TM also.

Solution to #4–cont'd

Example A — input 111111 (six 1s), start:

$\ldots \Delta \Delta \overset{\downarrow}{1} 1 1 1 1 1 \Delta \Delta \ldots$, where $\downarrow$ shows the initial head position.

1. Change leftmost 1 $\rightarrow$ *A* 1 1 1 1 1. Scan right and change rightmost unmarked 1 $\rightarrow$ *A* 1 1 1 1 *B*. Return left.
2. Change leftmost unmarked 1 (second cell) $\rightarrow$ *A A* 1 1 1 *B*. Change rightmost unmarked 1 (fifth cell) $\rightarrow$ *A A* 1 1 *B B*. Return left.
3. Change leftmost unmarked 1 (third cell) $\rightarrow$ *A A A* 1 *B B*. Change rightmost unmarked 1 (fourth cell) $\rightarrow$ *A A A B B B*. Return left.
4. Scan for an unmarked 1: none found (only *A*s/*B*s), so go to Cleanup.
5. Cleanup (convert $A \rightarrow 1$, $B \rightarrow \Delta$): final tape 1 1 1 (three 1s).

Under our encoding, six 1s means input number $x = k - 1 = 5$.
Output 111 encodes 2, and $\lfloor 5/2 \rfloor = 2$, which is correct.

Solution to #4–cont'd

Example B — input 11111 (five 1s), start: $\ldots \Delta\,\Delta\,\overset{\downarrow}{1}\,1\,1\,1\,1\,\Delta\Delta\ldots$, where $\downarrow$ shows the initial head position.

1. Change leftmost 1 $\to$ *A* 1 1 1 1. Scan right and change rightmost unmarked 1 $\to$ *A* 1 1 1 *B*. Return left.
2. Change next leftmost 1 (second cell) $\to$ *A A* 1 1 1 *B*. Change rightmost unmarked 1 to *B* (fifth cell) $\to$ *A A* 1 *B B*. Return left.
3. Mark leftmost unmarked 1 (middle cell) $\to$ *A A A B B*. Try to find a 1 on the right: none (hit Δ), so go to Cleanup.
4. Cleanup (convert *A* $\to$ 1, *B* $\to \Delta$): final tape 1 1 1 (three 1s).

Under our encoding, five 1s means input number $x = k - 1 = 4$. Output 111 encodes 2, and $\lfloor 4/2 \rfloor = 2$, which is correct.

Predicates are Turing Computable

- A *predicate* is a function that maps from any domain to the set of truth constants TRUE, FALSE.
- Logical propositions such as $x > y$, $x = y$, etc., can be seen as predicates $f : \mathbb{N} \times \mathbb{N} \rightarrow \{\text{TRUE}, \text{FALSE}\}$, and so on.
- Using the unary alphabet, predicates such as $x > y$ can be evaluated, with the convention that TRUE is 1 and FALSE is 0.
- As always, the inputs are on the tape separated by a Δ at the start, and upon completion, the TM should leave the output on the tape.

Examples of Predicates

- $<$, $>$, $\leq$, $\geq$, $\neq$ and such comparison operators are all predicates that can be computed by Turing Machines.
- Other predicates include divisibility $\textsc{div}(\cdot,\cdot)$ where $\textsc{div}(3,12)$ is $\textsc{true}$ and $\textsc{div}(3,10)$ is $\textsc{false}$), parity ($\textsc{even}(\cdot)$ and $\textsc{odd}(\cdot)$), etc.
- Predicates can also occur in conjunction with basic arithmetic functions (addition, multiplication).

Exercises

(5) Give a TM that computes the predicate $x > y$.

(6) Give a TM that computes the predicate $\textsc{odd}(x)$.

(7) Prove that the predicate $\textsc{div}(x+1, 2y)$ is Turing computable, by giving a TM that computes it.

(8) Let us define a *less than half* predicate $f : \mathbb{N} \times \mathbb{N} \to \{\textsc{true}, \textsc{false}\}$ by the following:

$$f(x, y) = \begin{cases} \textsc{true} & \text{if } 2x < y, \text{ and} \\ \textsc{false} & \text{otherwise.} \end{cases}$$

Prove that f is Turing computable by giving a TM that computes it.