Universal Turing Machine

Need for a Universal TM

- Each TM appears to be specialized at solving one particular problem. (Hardwired)
- Computers solve many problems → General purpose computers (Re-progammable)
- It is possible to invent a single TM which can be used to compute any computable sequence.

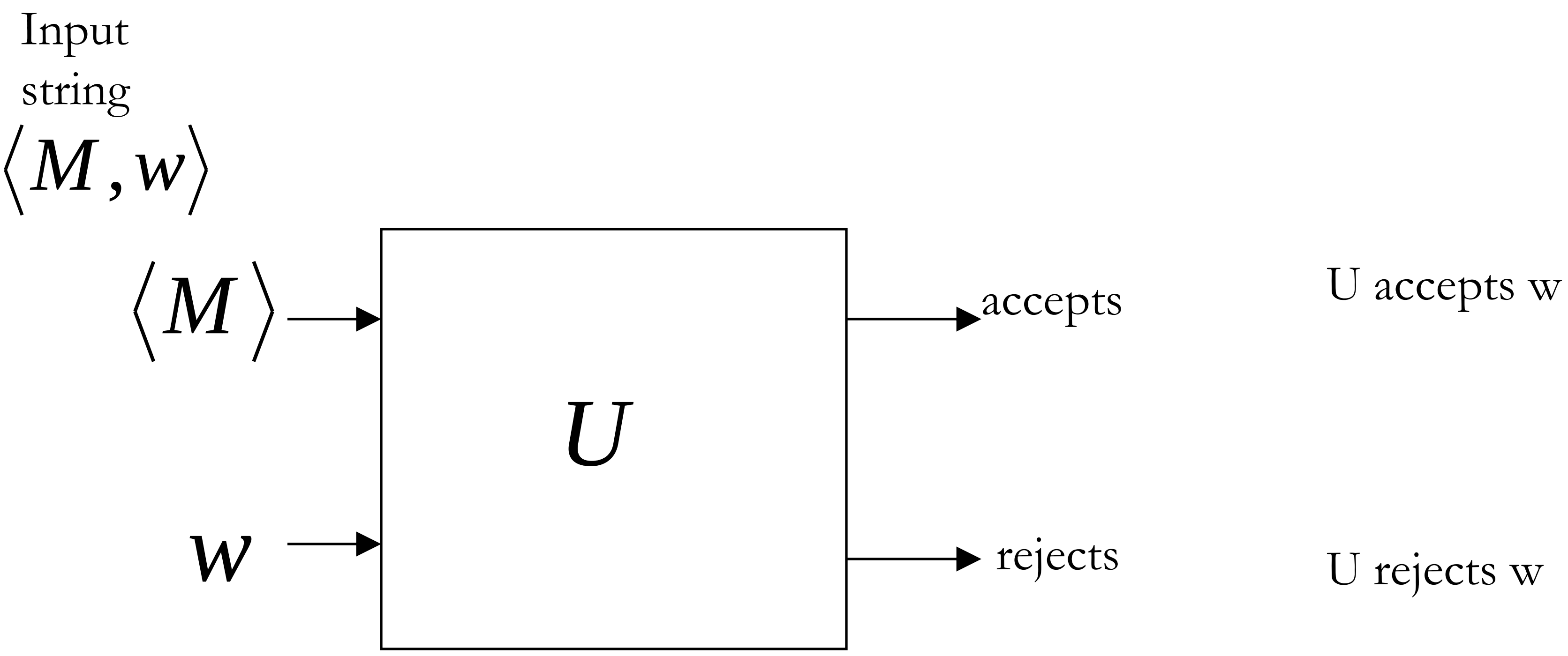
Universal TM

- A **universal Turing machine (UTM) is a Turing machine that can** simulate an arbitrary Turing machine on arbitrary input.
- The universal machine essentially achieves this by reading both the description of the machine to be simulated as well as the input thereof from its own tape.
- The UTM played an important early role in stimulating the development of stored-program computers.
- A universal TM can execute any algorithm, provided it receives an input string that describes the algorithm and any data it is to process.

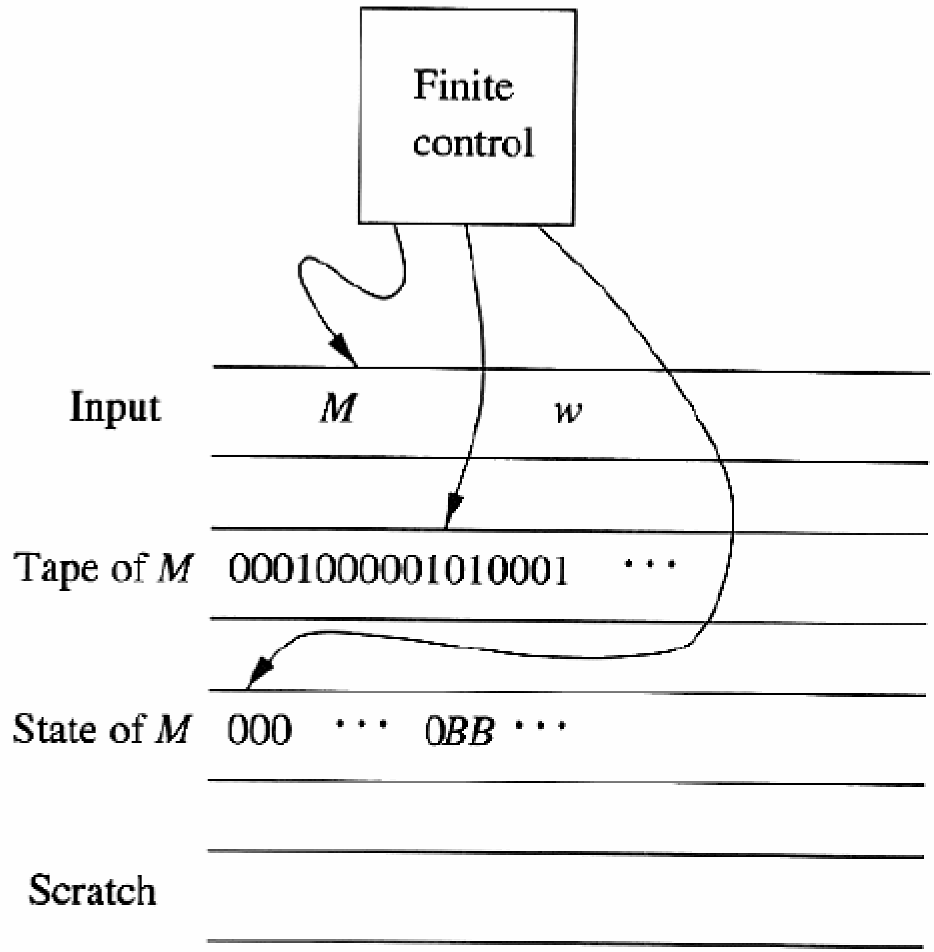
Definition

- A *UTM* is a Turing machine *U* that works as follows.
  - It is assumed to receive an input string of the form $e(M)e(w)$, where *M* is an arbitrary TM, *w* is a string over the input alphabet of *M* , and *e* is an encoding function whose values are strings in $\{0, 1\}^*$. The computation performed by *U* on this input string satisfies these two properties:
    1. *U* accepts the string $e(M)e(w)$ if and only if *M* accepts *w*.
    2. If *M* accepts *w* and produces output *y*, then *U* produces output $e(y)$.

Definition UTM

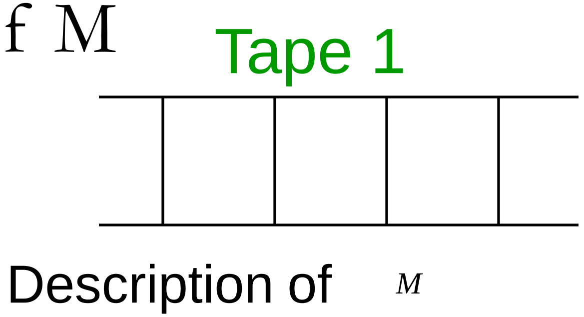


Organization of UTM



Organization of UTM

- Input of Universal Turing Machine U:
  - Description of transitions of M
  - Input string of M ie. w



- Describe Turing machine M as a string of symbols
- ie encode M as a string of symbols

Alphabet Encoding

Symbols:	a	b	c	d
Encoding:	0	00	000	0000

State Encoding

States:	q1	q2	q3	q4
	↓	↓	↓	↓
Encoding:	0	00	000	0000

Head Move Encoding

Move:	L	R
Encoding:	0	00

Transition Encoding

Transition:

$$\delta(q_1, a) = (q_2, b, L)$$

Encoding:

$$0\ 1\ 0\ 1\ 00\ 1\ 00\ 1\ 0$$

separator

Turing Machine Encoding

Transitions:

$$\delta(q_1, a) = (q_2, b, L) \qquad \delta(q_2, b) = (q_3, c, R)$$

Encoding:

0 1 0 1 00 1 00 1 0 11 0010010001 000100

separator

Tape 1 contents of Universal Turing Machine

Tape 1

0 1 0 1 00 1 00 1 0 11 0010010001 000100

- A Turing Machine is described with a binary string of 0's and 1's
- Therefore: The set of Turing machines forms a language L_u: each string of this language is the binary encoding of a Turing Machine

Language of Turing Machines

L = {010100101, TM1

00100100101111, TM2

111010011110010101,

...... }

- Define the language L_u as follows:

  L_u = {x | x is in {0, 1}* and x = <M,w> where *M* is a TM encoding and *w* is in L(M)}

Theorem: UTM exist

Proof. Define the *universal language* L_u as the set of all binary strings that encode pairs (M, x), where M is a Turing machine with binary input alphabet and x is a binary input string so that x lies in $L(M)$. Claim that there is a Turing machine U so that $L(U) = L_u$. Indeed, assume that U has multiple tapes. More precisely, the first tape initially holds the transitions of M, along with the string x. The second tape stores the simulated tape of M, and the third tape holds the state of M. The operations of U can be summarized as follows:

1. Examine the input to check whether the encoding of M is legitimate. If not, U halts without acceptance.
2. Initialize the second tape to contain the input string x in its encoded form (i.e., for each 0 in x place 10 on the tape and for each 1 in x place 100 there).

Theorem: UTM exist

3. Place 0, the start state of M, on the third tape, and move the head of U's second tape to the first simulated cell.
4. To simulate a transition of M, U searches on its first tape for a string $0^i10^j10^k10^l10^m$ so that 0^i is the state of the third tape, and 0^j is the tape symbol of M that begins at the position of the second tape. If so, U changes the contents of the third tape to 0^k, replaces 0^j on the second tape by 0^l, and keeps the head (N) on the second tape or moves the head on the second tape to the position of the next 1 to the left (L) or to the right (R).
5. If M has no transition that matches the simulated state and tape symbol, then in step 4, no transition will be found. Thus, M halts and U does likewise.
6. If M enters its accepting state, then U accepts (M, x).

In this way, U simulates M on x so that U accepts the encoded pair (M, x) if and only if M accepts x. This proves the claim. $\square$