50003 - Models Of Computation - (Prof Wiklicky) Lecture 4

Oliver Killane

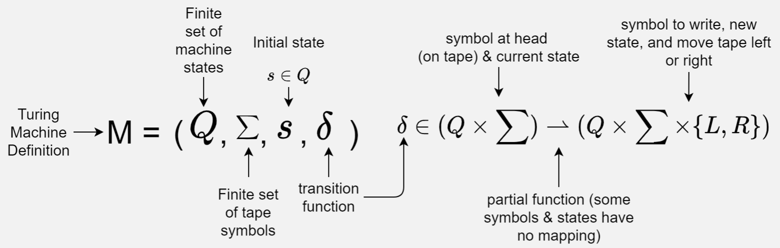
01/04/22

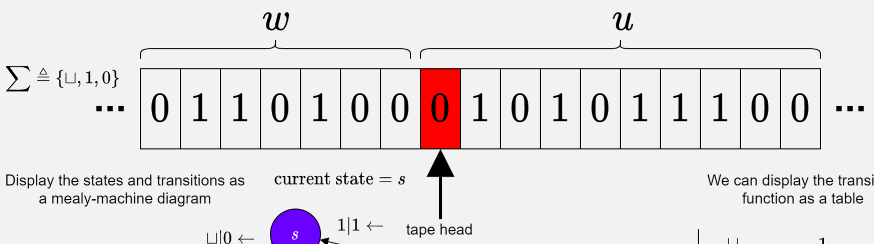
Lecture Recording

Lecture recording is available here

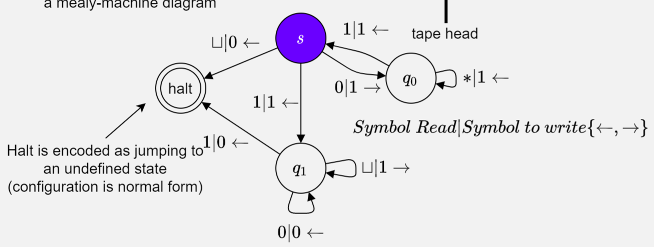
Turing Machines

Definition: Turing Machine





Display the states and transitions as a mealy-machine diagram

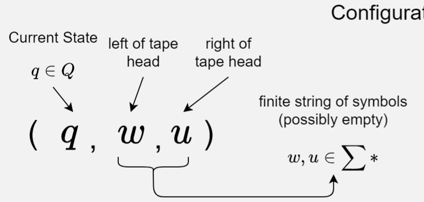


We can display the transition function as a table

	$\sqcup$	1	0
s		$(q_1, 1, L)$	$(q_0, 1, R)$
q_0	$(q_0, 1, L)$	$(q_0, 1, L)$	$(q_0, 1, L)$
q_1	$(q_1, 1, R)$		$(q_1, 0, L)$

No entry mean undefined, which results in a halt

Configuration:



Start State: With the start state, no symbols to the left and the input symbols to the right

(s, ϵ, u) where $\epsilon =$ empty

Normal form: when there is no transition for the current state and symbol at the tape head

(q, w, u) where $\delta(q, \mathit{first}\ u) \uparrow$

Transition:

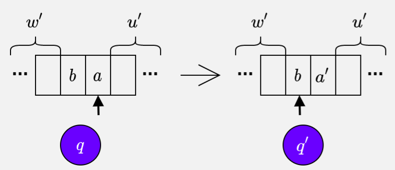
Note: When matching on an empty list, we get a new, empty element

$\sqcup : [\,] = [\,]$

$$(q, w, u) \rightarrow_M (q', w', u')$$

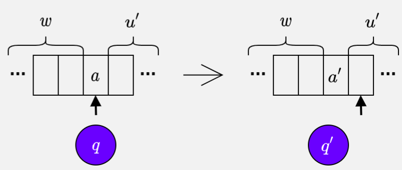
Transition moves to the left

$$\frac{a : u' = u,\ b : w' = w,\ \delta(q, a) = (q', a', L)}{(q, w, u) \rightarrow_M (q', w', b : a' : u')}$$



Transition moves to the right

$$\frac{a : u' = u,\ \delta(q, a) = (q', a', R)}{(q, w, u) \rightarrow_M (q', w : a', u')}$$



$$c_0 \rightarrow_M c_1 \rightarrow_M c_2 \rightarrow_M \cdots \rightarrow_M \begin{cases} c_n & \text{finite sequence - the turing machine halts} \\ \ldots & \text{infinite sequence - turing machine does not halt} \end{cases}$$

Register machines abstract away the representation of numbers and operations on numbers (just uses $\mathbb{N}$ with increment, decrement operations), **Turing machines** are a more concrete representation of computing.

Turing $\rightarrow$ Register Machine

We can show that any computation by a **Turing Machine** can be implemented by a **Register Machine**. Given a **Turing Machine** M:

1. Create a numerical encoding of M's finite number of states, tape symbols, and initial tape contents.
2. Implement the transition table as a register machine.
3. Implement a register machine program to repeatedly carry out $\rightarrow_M$

Hence **Turing Machine Computable** $\Rightarrow$ **Register Machine Computable**.

Turing Machine Number Lists

In order to take arguments, and return value we need to encode lists on number on the tape of a turing machine. This is done as strings of unary values.

$$\sum = \{\sqcup, 0, 1\} \quad \overbrace{\underbrace{\dots}_{\text{all } \sqcup}\ 0\ \underbrace{1\dots1}_{n_1}\ \sqcup\ \underbrace{1\dots1}_{n_2}\ \sqcup\ \dots\ \sqcup\ \underbrace{1\dots1}_{n_k}\ 0\ \underbrace{\dots}_{\text{all } \sqcup}}^{\text{Turing Machine Tape}}$$

Definition: Turing Computable

If $f : \mathbb{N}^n \rightharpoonup \mathbb{N}$ is **Turing Computable** iff there is a turing machine M such that:

From initial state $(s, \epsilon, [x_1, \dots, x_n])$ (tape head at the leftmost 0), M halts if and only if $f(x_1, \dots, x_n) \downarrow$, and halts with the tape containing a list, the first element of which is y such that $f(x_1, \dots, x_n) = y$.

More formally, given $M = (Q, \sum, s, \delta)$ to compute f:

$$f(x_1, \dots, x_n) \downarrow \wedge f(x_1, \dots, x_n) = y \Leftrightarrow (s, \epsilon, [x_1, \dots, x_n]) \rightarrow^*_M (*, \epsilon, [y, \dots])$$

Register $\rightarrow$ Turing Machine

It is also possible to simulate any register machine on a turing machine. As we can encode lists of numbers on the tape, we can simply implement the register machine operations as operations on integers on the tape.

Hence **Register Machine Computable** $\Rightarrow$ **Turing Machine Computable**.

Notions of Computability

Every computable algorithm can be expressed as a turing machine (**Church-Turing Thesis**). In fact **Turing Machines**, **Register Machines** and the **Lambda Calculus** are all equivalent (all determine what is computable).

- **Partial Recursive Functions** Godel and Kleene (1936)
- **λ-Calculus** Church (1936)
- **canonical systems for generating the theorems of a formal system** Post (1943) and Markov (1951)
- **Register Machines** Lambek and Minsky (1961)
- **And many more** (multi-tape turing machines, parallel computation, turing machines embeded in cellular automata etc))