Turing Machines and Equivalent Models

Deepak D'Souza

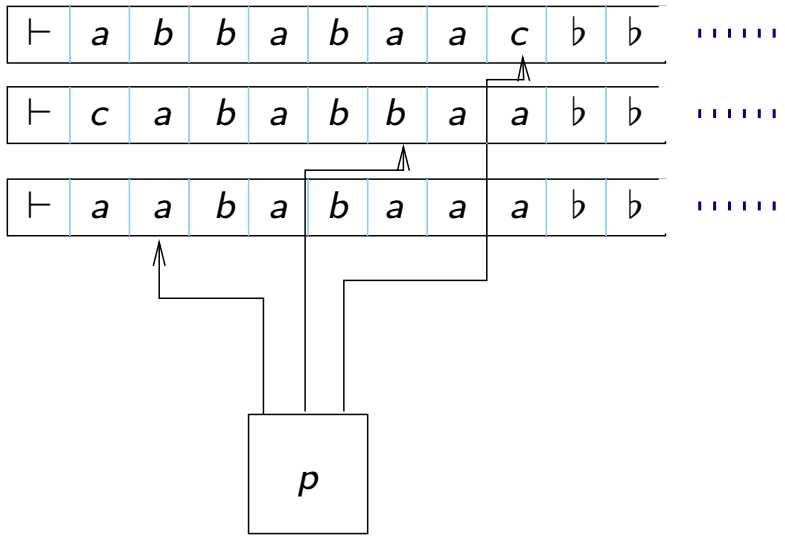
Indian Institute of Science,
Bangalore.

13 November 2019

Outline

1. Robustness of TM model
2. Other equivalent models

TM with multiple tapes



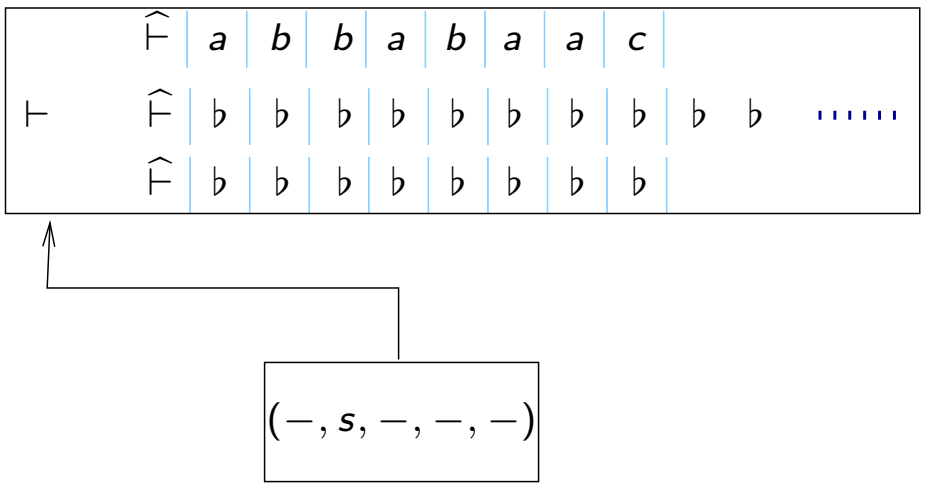
- Finite control
- Multiple tapes (say 3), each with its own read-write head.
- Each step: In current state p, read current symbols under the tape heads, say a, b, c: Change state to q, replace current symbols by a', b', c', and move each head left or right.

$$(p, a, b, c) \to (q, a', b', c', L/R, L/R, L/R).$$

How a TM can simulate a multi-tape TM

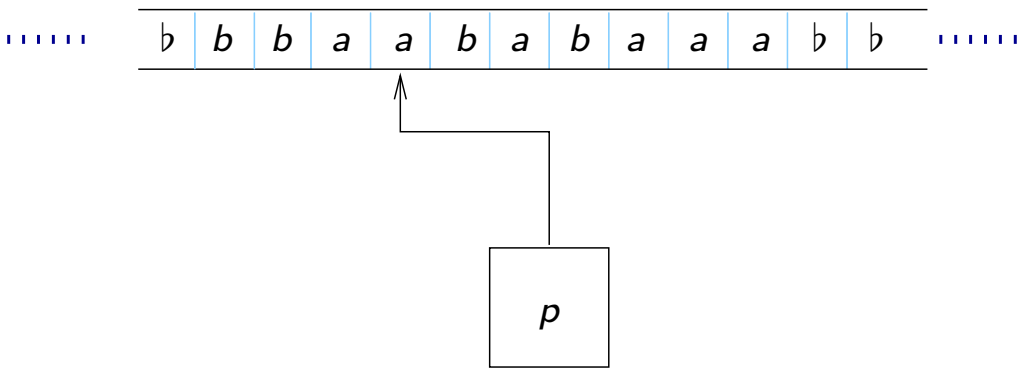
Let M be given multi-tape TM. Define a TM M' that:

- Given input w on its tape, first changes it to configuration:



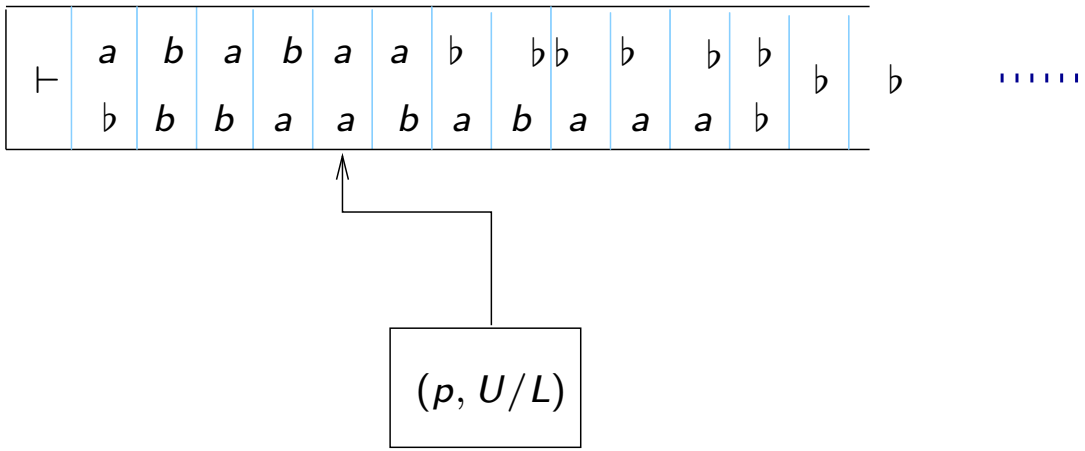
- Simulates a single step $(p, a, b, c) \to (q, a', b', c', L/R, L/R, L/R)$ of M by:
  - Scan top track to find $\widehat{a}$, and remember it in its finite control
  - Similarly scan tracks 2 and 3 and remember $\widehat{b}$ and $\widehat{c}$ in its finite control. State looks like $(-, p, a, b, c)$.
  - Now change to state $(-, q, a', b', c')$.
  - Scan track 1 to find $\widehat{a}$, replace it by a', move $\widehat{\ }$-mark L/R.
  - Similarly for tracks 2 and 3.

TM with two-way infinite tape



- Finite control
- Single two-way infinite tape.

Simulating a two-way infinite tape



- To simulate, imagine the tape is folded to the right at some point, and simulate with a tape alphabet $\Gamma \times \Gamma \cup \{\vdash, \flat\}$.

The Church-Turing Thesis

Church-Turing Thesis

The definition of computability based on Turing machines, captures the "right" notion of computability.

Turing computability coincides with several other notions of computability proposed based on different models, in the 1930's:

- Post systems (Emil Post)
- μ-recursive functions (Gödel, Herbrand)
- λ-calculus (Church, Kleene)
- Combinatory logic (Curry, Schönfinkel)

Non-deterministic TM

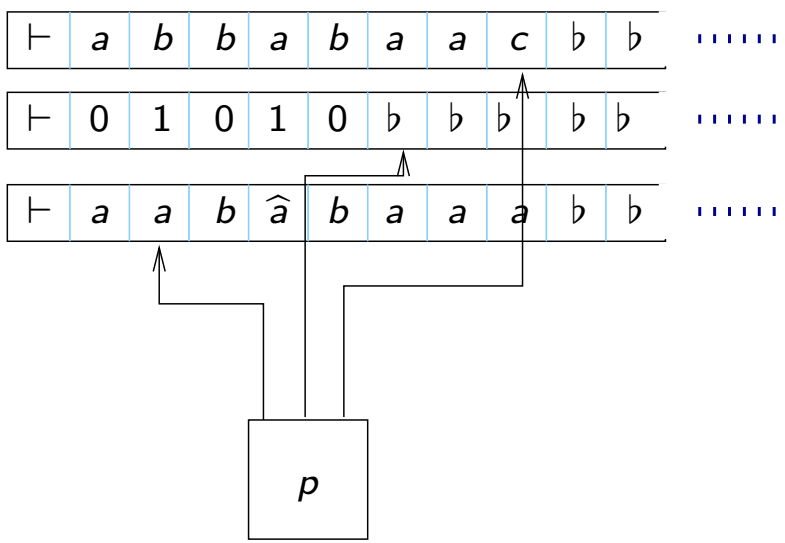
- Similar to TM already defined, except that moves can be non-deterministic.
- M accepts an input w if

$$(s, \vdash w\flat^{\omega}, 0) \stackrel{*}{\Rightarrow} (t, z, i),$$

for some z and i.

Simulating a non-deterministic TM by a det TM

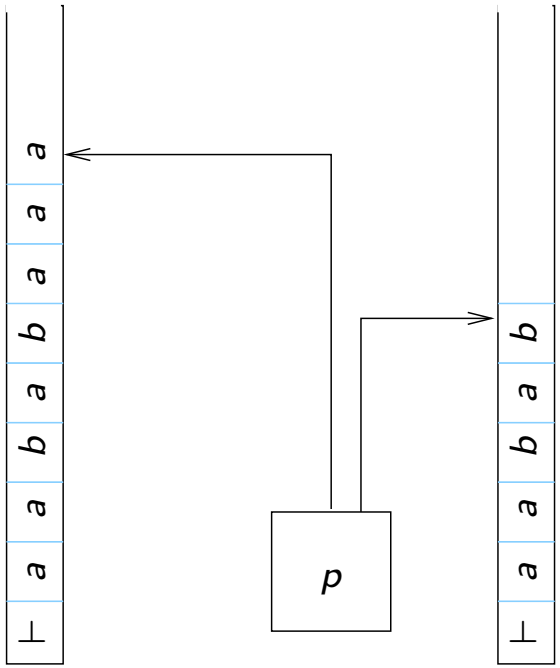
- Let M be a given non-det TM.
- Define a deterministic TM M' that accepts the same language as M.
- M' uses 3 tapes: for given input, guessed binary string, work tape to simulate run of M.



Simulating a non-deterministic TM by a det TM

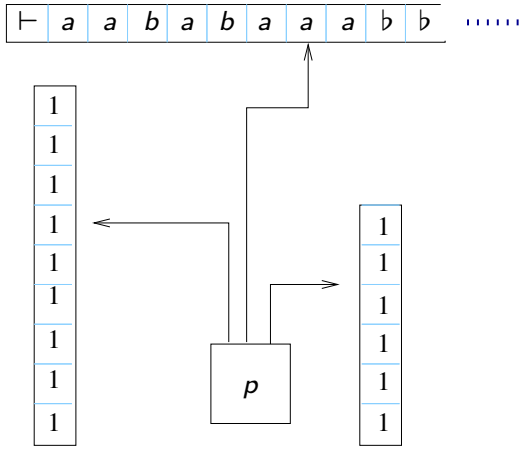
- Deterministic TM M' searches the tree representing the run of M on w, by doing a BFS on the tree.
- Enumerates binary strings in lexicographic order on tape 2.
- Simulate M on input along the path in tape 2, on tape 3.
- Accept (Enter state t) if M' enters t in the simulation.
- Guaranteed to capture all paths and accept iff the given non-deterministic TM accepts.

2-stack PDA



- The PDA can read the 2 top of stacks, and make a push/pop move independently on each stack.
- A 2-tape TM can easily simulate such a PDA.
- Conversely, a 2-stack PDA can simulate a TM.

Counter Machines



- In each step, based on the current symbol and its control state, a 2-counter machine (with counters c and d) can test c or d for zero, or increment/decrement c/d, move L/R, and change its control state.
- A TM can easily simulate a 2-counter machine.
- Conversely, a 4-counter machine can simulate a 2-stack PDA (and hence a TM).
- A 2-counter machine can simulate a 4-counter machine.