SOME COMPUTATION MODELS EQUIVALENT TO TURING MACHINES

Abhijit Das
Sudeshna Kolay

Department of Computer Science and Engineering
Indian Institute of Technology Kharagpur

March 15, 2022

Computational robustness of Turing Machines

- Standard model: Single semi-infinite tape with a two-way read/write head.
- We may try to enhance the performance of the standard model in several ways.
  - Add multiple tapes with independent two-way read/write heads.
  - Single tape but infinite in both directions.
  - A two-dimensional tape.
- Neither of these augmentations can improve the language-recognition capability of the standard model.
- We also investigate some restricted models.
  - 2-stack PDA
  - Counter automata
- These restricted machines are as powerful as standard TMs.
- The running times of the different models on the same input may vary widely.
- Enumeration machines are also equivalent to Turing machines. These machines justify the name *recursively enumerable*.

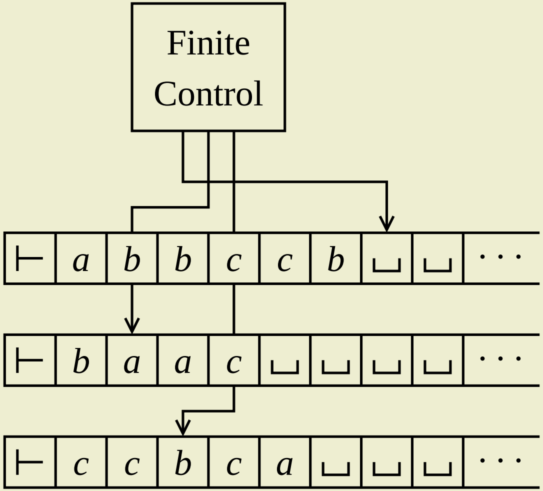
Allowing the head to stay stationary

- Transitions $\delta(p,a) = (q,b,S)$ are allowed.
- Replace this by two transitions:

$$\delta(p,a) = (tmp,b,R), \qquad \delta(tmp,A) = (q,A,L),$$

  Here, *tmp* is a temporary state, and A is any symbol of Γ.
- If needed, we will henceforth assume that a TM head may stay stationary during certain transitions.

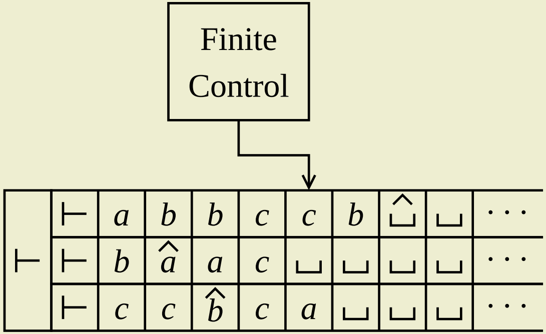
A 3-tape Turing machine M



The working of the 3-tape Turing machine M

- Let Γ (a superset of Σ) be the alphabet of each stack.
- There is only one set of states in the finite control.
- Initial configuration
  - The initial state is s.
  - The input is available at the beginning of the first tape (immediately after the left end marker).
  - The rest of the first tape, and all cells of the other two tapes contain the blank symbol (except for the leftmost cells).
  - All the three heads are positioned to the respective leftmost cells (and are scanning the left end markers).
- A transition is of the form $\delta(p, a_1, a_2, a_3) = (q, b_1, b_2, b_3, D_1, D_2, D_3)$.
- Acceptance is by entering the state t.
- Explicit rejection is by entering the state r.

Simulation of M by a standard 3-track TM N

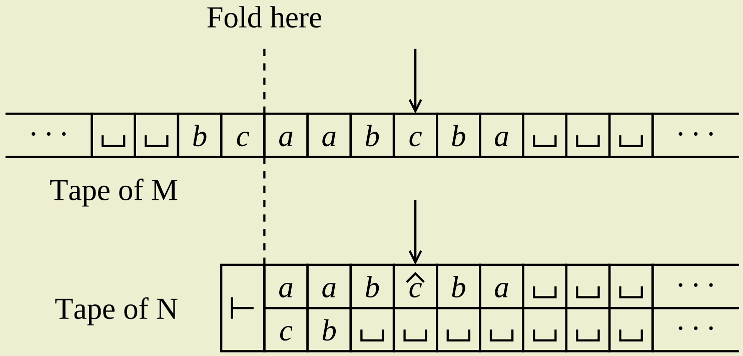

Finite
Control
$\vdash$
$\vdash$ a b b c c b $\hat{\sqcup}$ $\sqcup$ $\cdots$
$\vdash$ b $\hat{a}$ a c $\sqcup$ $\sqcup$ $\sqcup$ $\sqcup$ $\cdots$
$\vdash$ c c $\hat{b}$ c a $\sqcup$ $\sqcup$ $\sqcup$ $\cdots$

- Let $\hat{\Gamma} = \{\hat{a} \mid a \in \Gamma\}$.
- We take $\Gamma_N = \{\vdash\} \cup \left(\Gamma \cup \hat{\Gamma}\right)^3$.
- The blank for N is the blank of M in all the three tracks.

Simulation of M by N

- Initial configuration
  - The input is provided in the top track. The rest of the track and the other two tracks are empty.
  - The three left end markers have the hats to indicate the initial positions of the three heads of M.
- Simulation of a move of M
  - N scans its tape to find the hat on the first track. N remembers the track symbol in its finite control.
  - Likewise, N scans the hats on the second and the third tracks, and remembers the corresponding track symbols in its finite control.
  - The transitions of M are embedded in the finite control of N.
  - N relocates each of the three hats, replaces the hatted symbol by the unhatted symbols as M does, and moves each of the three hats left or right according to the transition of M.
- If M enters its accept/reject state, N also enters its accept/reject state.

Simulation of a 2-way semi-infinite tape by two tracks



- M has a 2-way read/write tape infinite in both directions.
- The input is given to M from some position.
- The head is initially positioned at the start symbol of the input.

Simulation of a 2-way semi-infinite tape by two tracks

- N folds the infinite tape into two tracks.
- The folding can be immediately before the start of the input.
- A hat represents the position of the head of M.
- N keeps on tracking the hat (no need to locate it).
- Depending upon the transition of M, N replaces the hatted symbol by the appropriate unhatted symbols, and moves the position of the hat to left or right on the same track, or from one track to the other if the folding position is crossed).

2-stack PDA P

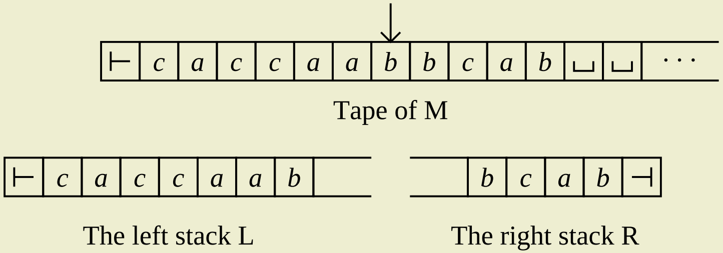
- (1-stack) PDA can accept only the context-free languages.
- A 2-stack PDA P works as follows.
  - The input is available on a one-way read-only input tape.
  - The input tape may be 2-way (but read-only). For our purpose, 1-way tapes suffice.
  - Each transition is of the form $\delta(p,a,A,B) = (q,\gamma,\delta)$ for $p,q \in Q$, $a \in \Sigma \cup \{\varepsilon\}$, $A,B \in \Gamma$, and $\gamma,\delta \in \Gamma^*$.
  - The acceptance may be by empty stack(s) and/or final state.
  - For the moment, assume that P is deterministic.
  - This restriction is not necessary. We will later see that non-deterministic TMs are as powerful as deterministic TMs.
- **Theorem:** A 2-stack PDA P is as powerful as a Turing machine M.

Simulation of P by M

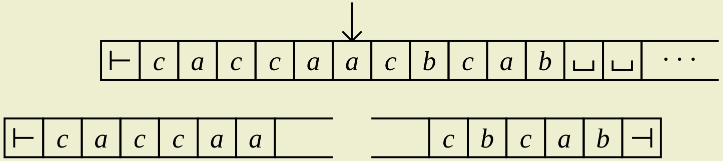
- M has three tapes.
- The input is provided on the first tape.
- The second tape simulates the first stack of P.
- The third tape simulates the second stack of P.
- Consuming a symbol from the input of P can be remembered by using a special marker on the symbols of the first tape.
- Replacing A by γ is a possibly multi-step process on the second tape.
- Replacing B by δ is a possibly multi-step process on the third tape.
- If P ever enters a final state and/or clears its stack after consuming the entire input, M enters t.
- M checks whether there are any unmarked symbols on the first tape to determine whether the entire input is read or not.

Simulation of *M* by *P*

(a) Storing the tape in two stacks



(b) Simulation of a move of M $\delta(p,b) = (q,c,L)$



Simulation of M by P

- The content of the tape of M to the left of the head (including the head position) is stored in the left stack L.
- The content of the tape of M to the right of the head (after the head position) is stored in the right stack R.
- The tape of M is semi-infinite.
- L is always a finite stack.
- R is an infinite stack.
- It suffices only to store the visited part of the tape, so R becomes finite.
- In order that R is never empty, an end marker always resides at the bottom of R. This marker indicates the end of the visited part on the tape of M, and may also be kept in the tape itself. Expansion of the visited part requires moving the marker to right.
- Since M never moves its head to the left of the end marker, L is never empty.

Simulation of M by P

- P first pushes the left end marker and all the symbols of the input to L.
- P then enters a loop that pops in L and pushes the popped symbol in R.
- The loop breaks when the left end marker is at the top of L.
- This creates the initial tape configuration of M in the two stacks of P.
- Subsequently, P keeps on making ε-transitions to simulate the moves of M.
- Each move of M involves popping a symbol from one stack and pushing a symbol to the other stack.
- The current state of M can be remembered in the finite control of P.
- If M enters t, P can clear its stack(s) and/or go to its final state.
- If M enters r, P may get stuck or enter an infinite loop in a non-final state.
- If M never halts, P happily keeps on simulating M for ever.

Counter automata

- Let k be a fixed positive integer.
- A k-counter automaton consists of:
  - A one-way (or two-way) read-only tape where the input is supplied.
  - k counters capable of storing non-negative integer values.
- Each counter can be independently operated, and supports the following operations.
  - Check for zero
  - Find the least significant bit of the counter value
  - Increment by 1
  - Decrement by 1 (does not apply when the counter stores 0)

Simulation of a stack by two counters C and D

- Let the stack alphabet be Γ.
- The symbols of Γ can be represented by $l = \lceil \log_2 |\Gamma| \rceil$ bits.
- We can think of the stack as storing bits only.
- Let (A) be the l-bit binary representation of $A \in \Gamma$.
- If the stack is intended to store $A_1 A_2 \ldots A_k \in \Gamma^*$ from bottom to top, then it actually stores the kl bits $(A_1)(A_2)\ldots(A_k)$ from bottom to top.
- The counter stores this bit sequence as the value $1(A_1)(A_2)\ldots(A_k)$ in the binary representation.
- Obtaining the top A_k of the stack involves popping and remembering l bits on the top of the stack.
- Pushing $A_{k+1} \in \Gamma$ is effected by pushing the l bits (A_{k+1}) to the top of the stack.

Pushing/Popping a bit b to the stack

- The counter C stores the content of the stack as an integer.
- The counter D stores 0.

push(b)

```
while (C != 0) {
   C--; D++; D++;
}
if (b == 1) D++;
```

pop( )

```
while (C != 0) {
   C--;
   if (C == 0) break;
   C--;
   D++;
}
```

- After a push or pop,
  - the value of D may be moved to C:

    ```
    while (D != 0){ D--; C++; }
    ```

  - or the roles of C and D may be interchanged.

Three counters suffice

- A TM can be simulated by two stacks.
- A stack can be simulated by two counters.
- So four counters can simulate the tape of a TM.
- For each stack, only one counter stores the content of the stack.
- The other counter helps in the implementation of the push and pop operations.
- We can use a single helping counter for the operations of both the stacks.
- Therefore only three counters suffice.
- So a 3-counter automaton is as powerful as a Turing machine.
- Since a push/pop operation takes time proportional to the value stored in the counter, the 3-counter automata are in general much slower than the TM it is simulating.
- But they are equivalent in terms of language recognition.

Two counters suffice

- Let the three counters store the values i, j, k.
- We store this triple uniquely in a single counter as the integer $2^i 3^j 5^k$.
- We use the second counter to do arithmetic on i, j, k.
- Example: Operations on j
  - Check for $j = 0$: Check for divisibility of $2^i 3^j 5^k$ by 3.
  - Find the least significant bit of $j > 0$: See the next slide
  - $j{+}{+}$: Multiply $2^i 3^j 5^k$ by 3.
  - $j{-}{-}$: Divide $2^i 3^j 5^k$ by 3 (allowed for $j > 0$).
- The 2-counter automaton is again much slower than the 3-counter automaton.
- But language-recognition capability does not decrease.

Find the least significant bit of j

- C stores $2^i 3^j 5^k$, and D stores 0. We want to find the parity of j.
- Divisibility by 3 of the value stored in C can be checked without destroying the value.
- Repeated division by 3 destroys the original value of j. We do not have a third counter to store j. Also, j is unbounded, and cannot be stored in the finite control. We can however store the parity of j (only one bit of information) in the finite control.

Convert $2^i 3^j 5^k$ to $2^i 7^j 5^k$

```
Remember the parity as 0 in the finite control
while (C stores a value divisible by 3) {
   while (C != 0) {
      Decrement C thrice
      Increment D seven times
   }
   while (D != 0) { D--; C++; } /* Or switch the roles of C and D */
   Flip the parity information in the finite control
}
```

- The parity of j is now stored in the finite control.
- Convert $2^i 7^j 5^k$ back to $2^i 3^j 5^k$ using a similar strategy.

Enumeration machines

- An enumeration machine E consists of the following.
    - A finite control with transitions akin to those of a TM.
    - A 2-way read/write **work tape**.
    - A 2-way write-only **output tape** where only the symbols of Σ can be written.
    - No accept or reject state.
    - A special **start state** s and a special **enumeration state** e.
- E starts in state s with both tapes empty.
- E proceeds as dictated by the transition function in the finite control.
- Some transitions let the machine write to the output tape.
- E may from time to time enter the state e. Whenever this happens,
    - the content of the output tape is enumerated, and
    - the output tape is erased, and the head is positioned to the beginning of this tape.
- E may keep on doing enumeration for ever.
- $\mathscr{L}(E)$ is the set of all strings over Σ, enumerated by E.

Simulation of an enumeration machine E by a Turing machine M

- M consists of three tapes (or tracks).
- On input x, M does the following:
  - M copies x to its third tape.
  - M uses its first tape as the work tape of E.
  - M uses its second tape as the output tape of E.
  - M follows the transitions of E embedded in its own finite control.
  - If (the simulation of) E ever enters the enumeration state e, M compares the content y of the second tape with the content x of the third tape.
  - If $y = x$, then M accepts by going to t.
  - If $y \neq x$, M continues the simulation of E (after erasing y in the second tape).
- M accepts x if and only if E enumerates x at some point of time.
- $\mathscr{L}(M) = \mathscr{L}(E)$.

Simulation of a Turing machine M by an enumeration machine E

- The work tape of E consists of two tracks.
  - The top track simulates the working of M on input strings.
  - The bottom track stores the input strings for M.
- E simulates M on **all** inputs over Σ.
- If a simulation lets M go to the accept state t, then
  - E copies the input from the bottom track of the work tape to the output tape, and
  - E enters the enumeration state e.
- A simulation of M on some input(s) x may never halt.
- If such a thing happens, E never gets a chance to simulate M on other inputs.
- So the simulations cannot proceed sequentially one after another.
- All the simulations must run simultaneously.

Timesharing by E

- Let $x_1, x_2, x_3, \ldots$ be a listing of all input strings listed in some order (Σ^* is countable).
- E can compute x_{n+1} from x_n.
- E maintains the bottom track of its work tape as $\#x_1\#x_2\#x_3\#\ldots\#x_n\#$.
- E maintains the top track of its work tape as $\#C_1\#C_2\#C_3\#\ldots\#C_n\#$, where C_i is the current configuration of M on input x_i. C_i stores the content of the tape (the visited part only), the head position, and the state of M.
- When E starts a new simulation on x_{n+1}, it appends $x_{n+1}\#$ to both the tracks.
- If some C_i expands, everything to the right is shifted to make room for this expansion.
- E simulates M
  - on x_1 for one step,
  - on x_1, x_2 for one step each,
  - on x_1, x_2, x_3 for one step each,
  - and so on.
- If M accepts some x_i, the simulation of E on x_i eventually encounters this.

Tutorial exercises

1. A Jump Turing machine (JTM) $J = (Q, \Sigma, \Gamma, \delta, \vdash, \sqcup, s, t, r)$ is like a standard one-tape Turing machine (TM) with the only exception that each transition of J is of the form $\delta(p, A) = (q, B, m)$, where $p, q \in Q$, and $A, B \in \Gamma$, and $m \in \mathbb{Z}$. This means that if the finite control of J is in the state p and the head of J scans the tape symbol A, then the state changes to q, the content of the tape cell is changed from A to B, and the head jumps by m cells relative to the current position. If $m = 0$, the head stays at the current cell. If $m > 0$, the head makes a right jump. If $m < 0$, the head makes a left jump with the understanding that if the head is at position i on the tape and $|m| > i$, then the head goes to the leftmost cell (which stores the left end-marker $\vdash$). Also assume that if A is $\vdash$, then $m \geqslant 0$. Prove that a JTM is equivalent to a TM.

2. Let M be a Turing machine with one semi-infinite tape and two read/write heads. Each transition of M is determined by the current state p of the finite control, and the two symbols a and b scanned by the two heads. A transition of M is of the form $\delta(p, a, b) = (q, c, d, D_1, D_2)$ implying that the finite control goes to state q, the symbol a at the cell pointed by the first head is replaced by c, and the symbol b at the cell pointed by the second head is replaced by d. If both the heads point to the same tape cell ($a = b$ in this case), then the symbol at this cell is replaced by c (not by d unless $c = d$). Finally, the first head moves by one cell in direction D_1 (left or right), and the second head moves by one cell in direction D_2. Argue that this two-head Turing machine M can be simulated by a standard Turing machine N with one semi-infinite tape and with only one read/write head.

Tutorial exercises

3. Consider a Turing Machine M with a two-dimensional tape. Cells in the zeroth row and in the zeroth column are marked by special end markers. The input is provided horizontally at the first row starting from the first column, the finite control of M is in the start state, and the head is positioned at the $(0,0)$-th cell of the tape. Each move of M is dependent on the state of the finite control and on the tape symbol scanned by the head. During each move, the finite control switches to a new state (which may be the same as the old state), the tape symbol currently scanned by the head is overwritten by a new symbol (which may be the same as the old symbol), and the head moves by one cell in one of the four directions: up, down, left, right. M accepts by entering an accept state. Show that $\mathscr{L}(M)$ is r.e.

4. A queue automaton is like a PDA with the only exception that the external memory is organized (and accessed) as a queue. Prove that queue automata are equivalent to Turing machines.