Modifying Turing Machines

COMPSCI 3331

Outline

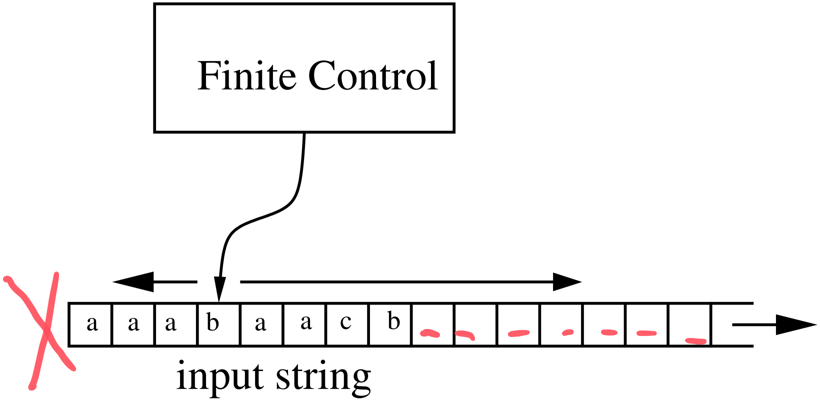
- Modifying TMs: restricted tapes, workspaces.
- Alternate Model: Type-0 Grammars.
- Church-Turing thesis.
- Nondeterministic TMs.

Modifying TMs

The power of TMs is not affected by minor changes in the TM model. For example:

- We can insist that the TM is a one-way infinite tape (i.e., has a starting point).
- We can allow the TM to have several tapes (work space).
- Nondeterminism is also OK.

One-way Infinite Tape

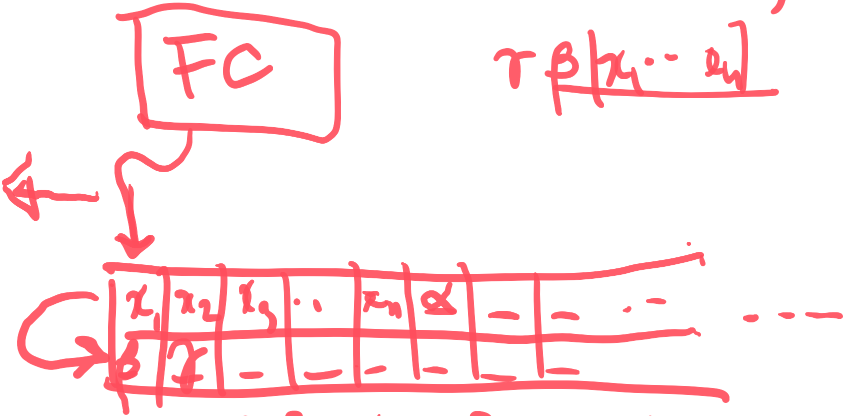


From Two-way to One-way Infinite Tape

IDEA: Replace tape alphabet Γ with Γ^2. Continue writing symbols on the left-infinite part (as necessary) of the tape underneath the right-infinite part.

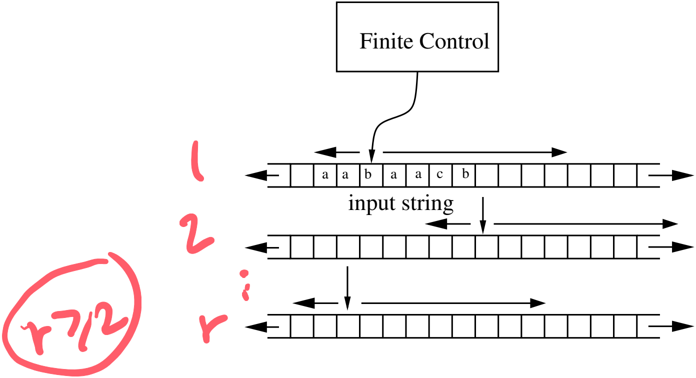
2-way

$\gamma \; \beta \; | \; x_1 \cdots x_n$



$\Gamma^2 = \Gamma \times \Gamma$

Multiple Tape TMs



- Input is always on the first tape.
- All other tapes are initially blank.

Action of a Multitape TM

At each step of a multitape TM:

- The state is updated.
- On each tape, the currently scanned symbol can be rewritten and the tape head moved (left, right or stationary).
- The tape heads can move independently: one head can move right, another left, etc.

Multitape TM Example

$L = \{a^n b^{\lfloor \sqrt{n} \rfloor} : n \geq 1\} = \{ab, aab, aaab, aaaabb, \ldots\}$.

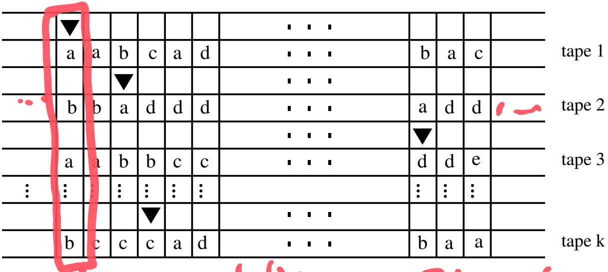
- Tape 1 $a^i b^j$ $i, j \geq 1$ if not, reject.
- copy b^j to tape 2
- write b^{j^2} on Tape 3
- add one b to Tape 2 $\to b^{j+1}$
- write $b^{(j+1)^2}$ on Tape 4
- compare num. of a's on tape 1 to the num of b's on Tapes 3 and 4. $j^2 \leq i \leq (j+1)^2$

ACCEPT if $j^2 \leq i$

Multitape TM to single-tape TM

IDEA: Simulate a k-tape TM by a single-tape TM with $2k$ 'tracks'.

extend alphabet



one letter on TM $(\Sigma \times \Gamma)^k$

- To simulate one step of the k-tape TM takes $O(m)$ time, where m is the length of the tape.
- Why? have to find each of the heads and simulate its action for one step.

Related Models

Even some models that are not TMs are equivalent to TMs:

- type-0 grammars.
- λ-calculus.

Type-0 Grammars

A type-0 grammar is a 4-tuple $G = (V, \Sigma, P, S)$ where

- V is a finite set of non-terminals.
- Σ is a finite alphabet.
- S is a distinguished start symbol.
- P is a finite set of productions of the form

$$\alpha \to \beta$$

where $\alpha, \beta \in (V \cup \Sigma)^*$ and $\alpha \neq \varepsilon$.

A word $w \in \Sigma^*$ is generated by G iff $S \Rightarrow^* w$.

$A \to \alpha$

"context free"

Type-0 Grammars

Example (Hopcroft and Ullman 1979, p. 220):

$$
\begin{array}{rcl|rcl}
S & \to & ACaB & aD & \to & Da \\
Ca & \to & aaC & AD & \to & AC \\
CB & \to & DB & aE & \to & Ea \\
CB & \to & E & AE & \to & \varepsilon
\end{array}
$$

$$L(G) = \{a^{2^n} \ : \ n \geq 1\}.$$

Thm. The class of languages generated by type-0 grammars are exactly the class of languages recognized by TMs.

$$S \Rightarrow ACaB \Rightarrow AaaCB \Rightarrow AaaE \stackrel{*}{\Rightarrow} AEaa \Rightarrow \varepsilon aa = a^2$$

Church-Turing Thesis

- The Church-Turing thesis states that the TMs capture our notion of what is computable.
- Any of the models we prove are equivalent to TMs are also considered **universal** models of computation.
- Church proposed another universal model of computation: λ-calculus.

Computers and TMs

Simulating a TM on a computer:

- Encode states of the TM as strings.
- Create a lookup table of the transition of the TM.
- Simulate the transitions directly.

Simulating a computer with a TM:

- The TM simulates machine code execution: it stores all the information we need to execute this code (PC, registers, separate tapes for code, memory, stack, etc.)

Nondeterministic TMs

TMs are **deterministic** by nature. We can also define nondeterministic TMs. In this case, $\delta : Q \times \Gamma \rightarrow 2^{Q \times \Gamma \times \{L,R,S\}}$.

- $\delta(q, \alpha) = \{(q_1, \beta_1, D_1), \ldots, (q_n, \beta_n, D_n)\}$ for some $n \geq 0$.
- We can choose any transition to apply. We accept if there is any accepting path.

Nondeterministic TMs

Thm. Let M be a nondeterministic TM. Then there exists a deterministic TM M' which accepts the same language.

Proof. Our TM M' performs a breadth-first search of all possible paths that M' could go down.

$\Gamma^* Q \Gamma^*$

- We store a list of IDs of M on tape 1 of M'.
- We will use other tapes of M' to update the list of IDs.
- Initially, tape 1 contains the start ID: $q_0 x$, where x is the input word.
- We then process each ID $w_1 q w_2$ on tape 1 in turn.
- If $w_1 q w_2 \vdash_M w_1' q' w_2'$, then we add $w_1' q' w_2'$ to tape 1 of M'.

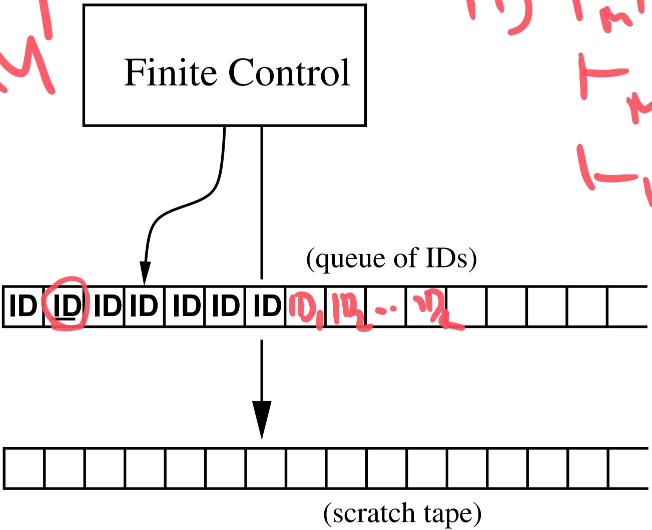
nondeterministic TM

Nondeterministic TMs

M'

$ID \vdash_M ID_1 \vdash_M ID_2 \vdash_M ID_k$

tape 1



Nondeterministic TMs

(det)

- If M' finds an accepting ID of M on tape 1, then M' accepts.
- In this way, M' only accepts words that M accepts.
- If M accepts, then M' will eventually find the accepting path.
- This is because each ID can only have a finite number of IDs that can come after it. ($2^{3|Q||\Gamma|}$)

Where to from here?

- We know how TMs function.
- We know that many different models that are equivalent to TMs.
- How can we describe the languages that can be **accepted** by a TM?