CSCE 355
4/3/2024

Project FAQs

ε-removal ← ε-NFA ~~(standard input(?))~~ from a file?

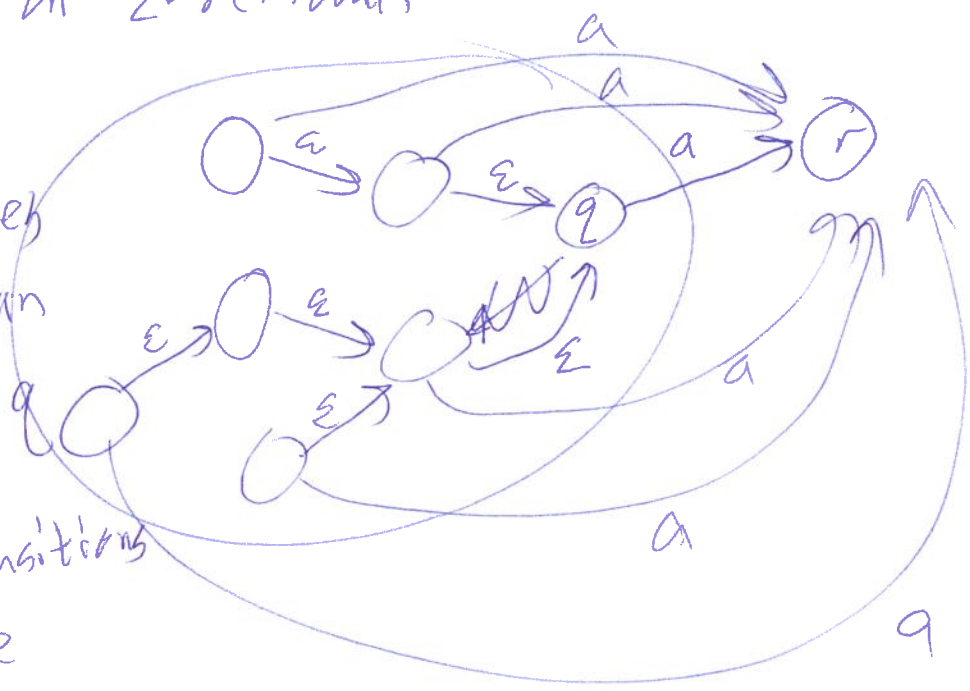
→ NFA (standard output)

simulator ← NFA file

← strings from std input

→ standard output

Hint on ε-removal:



list of states that can reach q via ε-transitions alone

[pre-compute this list for each q]

Turing Machines (cont.)

Last time: A TM is a tuple (blank symbol)

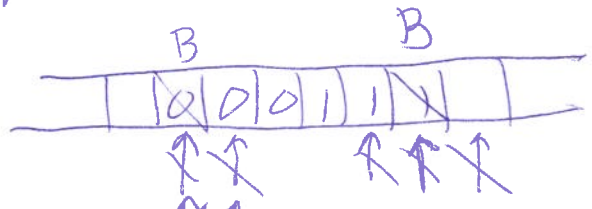
$$(Q, \Sigma, \Gamma, \delta, q_0, B, F)$$

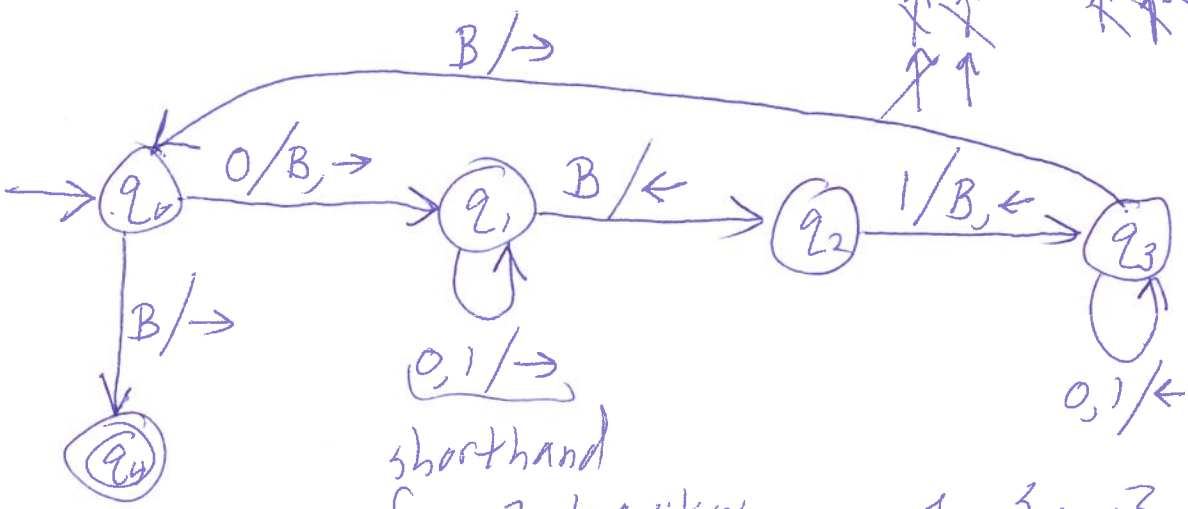
- Q: state set
- Σ: input alphabet
- Γ: tape alphabet
- δ: partial transition function
- q_0: start state
- B: blank symbol
- F: set of accepting states

$$\delta : Q \times \Gamma \to Q \times \Gamma \times \{\leftarrow, \rightarrow\} \quad \text{partial}$$

- Q: current state
- Γ: symbol scanned
- Q: new state
- Γ: scanned symbol replacement
- $\{\leftarrow, \rightarrow\}$: head movement

Ex: $L = \{0^n 1^n : n \geq 0\}$





shorthand for 2 transitions:

0/0,→
1/1,→

$\Sigma = \{0, 1\}$

$\Gamma = \{0, 1, B\}$

Missing transitions correspond to undefined $\delta(\,,\,)$ output

Tabular form

	0	1	B
$\rightarrow q_0$	$(q_1, B, \rightarrow)$	—	$(q_4, B, \rightarrow)$
q_1	$(q_1, 0, \rightarrow)$	$(q_1, 1, \rightarrow)$	$(q_2, B, \leftarrow)$
q_2	—	$(q_3, B, \leftarrow)$	—
q_3	$(q_3, 0, \leftarrow)$	$(q_3, 1, \leftarrow)$	$(q_0, B, \rightarrow)$
$*\, q_4$	—	—	—

"—" means "undefined"

~~Examples~~

01 accept

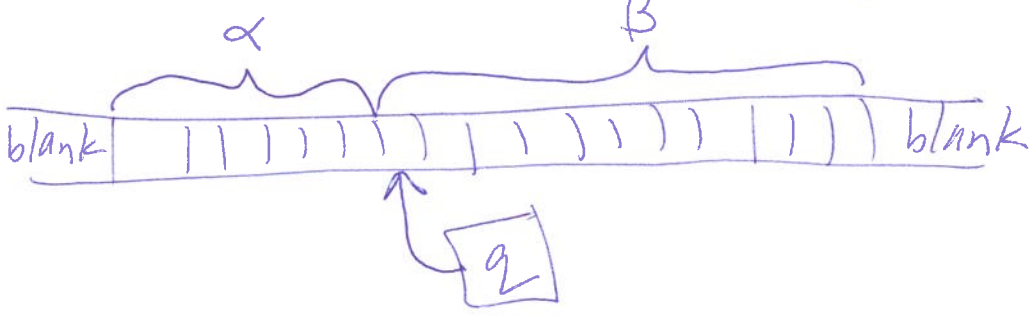
10 reject (no transition out of q_0 on 1)

~~000~~ reject (no transition out of q_2 on 1)

Def: Let $M := \langle Q, \Sigma, \Gamma, \delta, q_0, B, F \rangle$ be a TM. An ID (or configuration) of M is a string of the form

$$\alpha q \beta \qquad (\alpha, \beta \in \Gamma^*, q \in Q)$$

$\beta \neq \varepsilon$



means: (1) $\alpha\beta$ is a contiguous segment of the tape long enough to include:
- all nonblank symbols on the tape
- the cell being scanned

(2) M is currently in state q

(3) M is scanning the first symbol of β.

Let ID_1 & ID_2 be 2 IDs of M.

We say that $ID_1 \vdash ID_2$ (ID_2 is the one-step successor of ID_1), formally,

let $ID_1 := \alpha q a \gamma$. $\alpha, \gamma \in \Gamma^*$, $a \in \Gamma$, $q \in Q$

Then $ID_1 \vdash ID_2$ iff

1) $\delta(q,a) = (r, b, \rightarrow)$ and $ID_2 = \alpha b r \gamma$

2) $\delta(q,a) = (r, b, \leftarrow)$ and

$ID_2 = \alpha' r c b \gamma$ where $\alpha = \alpha' c$

[3) If $\delta(q,a)$ is undefined, then ID_1 has no successor

ID padding: Can freely pad an ID on the left or right with blank symbols, & we consider this to be the same ID.

Def: Let M be a TM as above and w an input string. The initial ID of M on input w is $q_0 w$

Def: M, w as in the prev definition. The computation of M on w is the sequence of IDs

$$ID_0 \vdash ID_1 \vdash \cdots$$

where ID_0 is the initial ID of M on input w, and $ID_i \vdash ID_{i+1}$ for $i = 0, 1, 2, \ldots$

~~This~~ The computation can be finite or infinite. It is finite if some ID_k in the sequence has no successor. ID_k is then the halting ID. If this is the case, then ID_k is accepting if its state is in F; else rejecting.

- M accepts w if its computation on w is finite, ending with an accepting ID.
- M rejects w if its computation on w is finite, ending with a rejecting ID.
- M loops on w if its computation on w is infinite.

$L(M)$ [the lang recognized by M] is
" " of M

$$\{w \in \Sigma^* : M \text{ accepts } w\}$$

M recognizes L if $L = L(M)$.

Def: M is a decider (or is total) if M halts on all input strings.

If M is a decider, then we also say $L(M)$ is decided by M.

A lang. L is Turing-recognizable (T-rec), if $L = L(M)$ for some TM M. L is decidable if $L = L(M)$

for some decider M.

Clearly decidable $\Rightarrow$ T-rec.

$\not\Leftarrow$ [we will show this]

Church–Turing thesis: Any reasonable model of general-purpose computation is equivalent to the TM model.