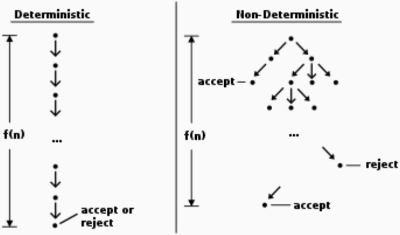
Pre-lecture brain teaser

So far we've only discussed deterministic Turing machines. However, similar to the relationship between DFAs and NFAs, there exists non-determistic Turing computation follows a non-deterministic path. So based on your knowledge of DFAs/NFAs and Turing machines I have two questions:



- What does a non-deterministic Turing machine look like?
- What languages are accept by non-deterministic Turing machines?

CS/ECE-374: Lecture 10 - Midterm 1 Review

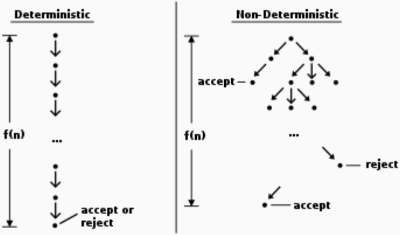
Lecturer: Nickvash Kani
Chat moderator: Samir Khan

February 25, 2021

University of Illinois at Urbana-Champaign

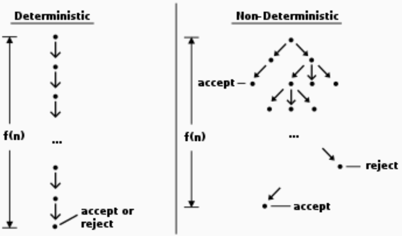
Pre-lecture brain teaser

So far we've only discussed deterministic Turing machines. However, similar to the relationship between DFAs and NFAs, there exists non-determistic Turing computation follows a non-deterministic path. So based on your knowledge of DFAs/NFAs and Turing machines I have two questions:



Pre-lecture brain teaser

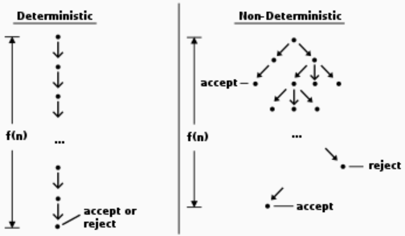
So far we've only discussed deterministic Turing machines. However, similar to the relationship between DFAs and NFAs, there exists non-determistic Turing computation follows a non-deterministic path. So based on your knowledge of DFAs/NFAs and Turing machines I have two questions:



- What does a non-deterministic Turing machine look like?

Pre-lecture brain teaser

So far we've only discussed deterministic Turing machines. However, similar to the relationship between DFAs and NFAs, there exists non-determistic Turing computation follows a non-deterministic path. So based on your knowledge of DFAs/NFAs and Turing machines I have two questions:



- What does a non-deterministic Turing machine look like?
- What languages are accept by non-deterministic Turing machines?

Exam Content

Including but not limited to:

- Languages and strings
- Regular expressions
- Deterministic finite automata
- Non-deterministic finite automata
- Equivalence of DFAs/NFAs/RegEx
- Regular language closure properties
- Fooling Sets

Strings

String Definitions

Definition

1. A string/word over Σ is a **finite sequence** of symbols over Σ. For example, '0101001', '*string*', '$\langle \text{moveback} \rangle \langle \text{rotate90} \rangle$'
2. ϵ is the empty string.
3. The length of a string w (denoted by $|w|$) is the number of symbols in w. For example, $|101| = 3$, $|\epsilon| = 0$
4. For integer $n \geq 0$, Σ^n is set of all strings over Σ of length n. Σ^* is the set of all strings over Σ.
5. *concatenation* defined recursively :
   - $xy = y$ if $x = \epsilon$
   - $xy = a(wy)$ if $x = aw$

Induction on strings

Inductive proofs on strings

Inductive proofs on strings and related problems follow inductive definitions.

Definition
The reverse w^R of a string w is defined as follows:

- $w^R = \epsilon$ if $w = \epsilon$
- $w^R = x^R a$ if $w = ax$ for some $a \in \Sigma$ and string x

Inductive proofs on strings

Inductive proofs on strings and related problems follow inductive definitions.

Definition
The reverse w^R of a string w is defined as follows:

- $w^R = \epsilon$ if $w = \epsilon$
- $w^R = x^R a$ if $w = ax$ for some $a \in \Sigma$ and string x

Theorem
Prove that for any strings $u, v \in \Sigma^*$, $(uv)^R = v^R u^R$.

Example: $(dog \bullet cat)^R = (cat)^R \bullet (dog)^R = tacgod.$

Principle of mathematical induction

Induction is a way to prove statements of the form $\forall n \geq 0, P(n)$ where $P(n)$ is a statement that holds for integer n.

Example: Prove that $\sum_{i=0}^{n} i = n(n+1)/2$ for all n.

Induction template:

- **Base case:** Prove $P(0)$
- **Induction hypothesis:** Let $k > 0$ be an **arbitrary** integer. Assume that $P(n)$ holds for any $n \leq k$.
- **Induction Step:** Prove that $P(n)$ holds, for $n = k + 1$.

By induction on |u|

Theorem
Prove that for any strings $u, v \in \Sigma^*$, $(uv)^R = v^R u^R$.

Proof by induction on $|u|$ means that we are proving the following.

Base case: Let u be an arbitrary string of length 0. $u = \epsilon$ since there is only one such string. Then

$(uv)^R = (\epsilon v)^R = v^R = v^R \epsilon = v^R \epsilon^R = v^R u^R$

By induction on $|u|$

Theorem
Prove that for any strings $u, v \in \Sigma^*$, $(uv)^R = v^R u^R$.

Proof by induction on $|u|$ means that we are proving the following.

Base case: Let u be an arbitrary string of length 0. $u = \epsilon$ since there is only one such string. Then

$$(uv)^R = (\epsilon v)^R = v^R = v^R \epsilon = v^R \epsilon^R = v^R u^R$$

Induction hypothesis: $\forall n \geq 0$, for any string u of length n:

For all strings $v \in \Sigma^*$, $(uv)^R = v^R u^R$.

By induction on |u|

Theorem
Prove that for any strings $u, v \in \Sigma^*$, $(uv)^R = v^R u^R$.

Proof by induction on $|u|$ means that we are proving the following.

Base case: Let u be an arbitrary string of length 0. $u = \epsilon$ since there is only one such string. Then

$$(uv)^R = (\epsilon v)^R = v^R = v^R\epsilon = v^R\epsilon^R = v^R u^R$$

Induction hypothesis: $\forall n \geq 0$, for any string u of length n:

For all strings $v \in \Sigma^*$, $(uv)^R = v^R u^R$.

No assumption about v, hence statement holds for all $v \in \Sigma^*$.

Inductive step

- Let u be an arbitrary string of length $n > 0$. Assume inductive hypothesis holds for all strings w of length $< n$.
- Since $|u| = n > 0$ we have $u = ay$ for some string y with $|y| < n$ and $a \in \Sigma$.
- Then

Inductive step

- Let u be an arbitrary string of length $n > 0$. Assume inductive hypothesis holds for all strings w of length $< n$.
- Since $|u| = n > 0$ we have $u = ay$ for some string y with $|y| < n$ and $a \in \Sigma$.
- Then

$$(uv)^R \quad =$$

Inductive step

- Let u be an arbitrary string of length $n > 0$. Assume inductive hypothesis holds for all strings w of length $< n$.
- Since $|u| = n > 0$ we have $u = ay$ for some string y with $|y| < n$ and $a \in \Sigma$.
- Then

$$\begin{aligned}
(uv)^R &= ((ay)v)^R \\
&= (a(yv))^R \\
&= (yv)^R a^R \\
&= (v^R y^R) a^R \\
&= v^R (y^R a^R) \\
&= v^R (ay)^R \\
&= v^R u^R
\end{aligned}$$

Induction on $|v|$

Theorem
Prove that for any strings $u, v \in \Sigma^*$, $(uv)^R = v^R u^R$.

Proof by induction on $|v|$ means that we are proving the following.

Induction on $|v|$

Theorem
Prove that for any strings $u, v \in \Sigma^*$, $(uv)^R = v^R u^R$.

Proof by induction on $|v|$ means that we are proving the following.

Induction hypothesis: $\forall n \geq 0$, for any string v of length n:

For all strings $u \in \Sigma^*$, $(uv)^R = v^R u^R$.

Induction on $|v|$

Theorem
Prove that for any strings $u, v \in \Sigma^*$, $(uv)^R = v^R u^R$.

Proof by induction on $|v|$ means that we are proving the following.

Induction hypothesis: $\forall n \geq 0$, for any string v of length n:

For all strings $u \in \Sigma^*$, $(uv)^R = v^R u^R$.

Base case: Let v be an arbitrary string of length 0. $v = \epsilon$ since there is only one such string. Then

$$(uv)^R = (u\epsilon)^R = u^R = \epsilon u^R = \epsilon^R u^R = v^R u^R$$

Inductive step

- Let v be an arbitrary string of length $n > 0$. Assume inductive hypothesis holds for all strings w of length $< n$.
- Since $|v| = n > 0$ we have $v = ay$ for some string y with $|y| < n$ and $a \in \Sigma$.
- Then

$$\begin{aligned} (uv)^R &= (u(ay))^R \\ &= ((ua)y)^R \\ &= y^R(ua)^R \\ &= ?? \end{aligned}$$

Inductive step

- Let v be an arbitrary string of length $n > 0$. Assume inductive hypothesis holds for all strings w of length $< n$.
- Since $|v| = n > 0$ we have $v = ay$ for some string y with $|y| < n$ and $a \in \Sigma$.
- Then

$$\begin{aligned}(uv)^R &= (u(ay))^R \\ &= ((ua)y)^R \\ &= y^R(ua)^R \\ &= ??\end{aligned}$$

Cannot simplify $(ua)^R$ using inductive hypothesis. Can simplify if we extend base case to include $n = 0$ **and** $n = 1$. However, $n = 1$ itself requires induction on $|u|$!

Regular expressions

Inductive Definition

A regular expression **r** over an alphabet Σ is one of the following:

Base cases:

- $\emptyset$ denotes the language $\emptyset$
- ϵ denotes the language $\{\epsilon\}$.
- a denote the language $\{a\}$.

Inductive Definition

A regular expression $\mathbf{r}$ over an alphabet Σ is one of the following:

Base cases:

- $\emptyset$ denotes the language $\emptyset$
- ϵ denotes the language $\{\epsilon\}$.
- a denote the language $\{a\}$.

Inductive cases: If $\mathbf{r_1}$ and $\mathbf{r_2}$ are regular expressions denoting languages R_1 and R_2 respectively then,

- $(\mathbf{r_1} + \mathbf{r_2})$ denotes the language $R_1 \cup R_2$
- $(\mathbf{r_1} \bullet \mathbf{r_2}) = r_1 \bullet r_2 = (\mathbf{r_1}\mathbf{r_2})$ denotes the language $R_1 R_2$
- $(\mathbf{r_1})^*$ denotes the language R_1^*

Regular Languages vs Regular Expressions

Regular Languages	Regular Expressions
$\emptyset$ regular	$\emptyset$ denotes $\emptyset$
$\{\epsilon\}$ regular	ϵ denotes $\{\epsilon\}$
$\{a\}$ regular for $a \in \Sigma$	$\mathbf{a}$ denote $\{a\}$
$R_1 \cup R_2$ regular if both are	$\mathbf{r_1} + \mathbf{r_2}$ denotes $R_1 \cup R_2$
R_1R_2 regular if both are	$\mathbf{r_1} \bullet \mathbf{r_2}$ denotes R_1R_2
R^* is regular if R is	$\mathbf{r}^*$ denote R^*

Regular expressions denote regular languages — they explicitly show the operations that were used to form the language

Practice Problem [True/False]

The language $\left\{ 0^i 1^j 0^k 1^\ell \;\middle|\; i, j, k, \ell \geq 0 \right\}$ is not regular.

Practice Problem

What is the regular expression for:

- All strings except 11.

Practice Problem

What is the regular expression for:

- All strings except 11.
- All strings that do *not* contain 000 as a subsequence.

Deterministic finite automata

Formal Tuple Notation

Definition
A deterministic finite automata (DFA) $M = (Q, \Sigma, \delta, s, A)$ is a five tuple where

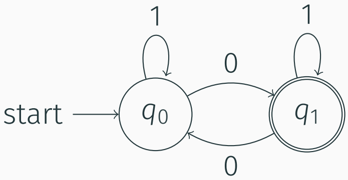
- Q is a finite set whose elements are called states,
- Σ is a finite set called the input alphabet,
- $\delta : Q \times \Sigma \to Q$ is the transition function,
- $s \in Q$ is the start state,
- $A \subseteq Q$ is the set of accepting/final states.

Common alternate notation: q_0 for start state, F for final states.

DFA Notation

$$M = \Big(\overbrace{Q}^{} \;,\; \underbrace{\Sigma}_{} \;,\; \overbrace{\delta}^{} \;,\; \underbrace{s}_{} \;,\; \overbrace{A}^{} \Big)$$

Example



- $Q =$
- $\Sigma =$
- $\delta =$
- $s =$
- $A =$

Practice Problem

Draw the DFA representing the regular language:
$L = \left\{ 0^i 1^j 0^k 1^\ell \mid i, j, k, \ell \geq 0 \right\}$

Non-deterministic Finite automata

Formal definition of language accepted by M

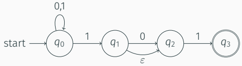
Definition
The language $L(M)$ accepted by a DFA $M = (Q, \Sigma, \delta, s, A)$ is

$$\{w \in \Sigma^* \mid \delta^*(s, w) \in A\}.$$

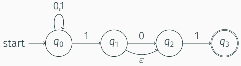
Another Way to look at NFAs

Is 010101 accepted?

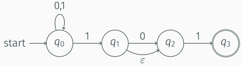
0



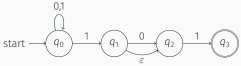
1



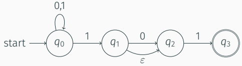
0



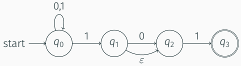
1



0

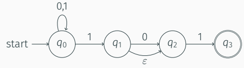


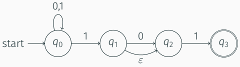
1

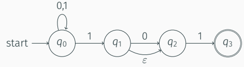


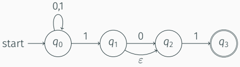
Another Way to look at NFAs

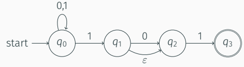
Is 010100 accepted?

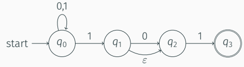
0


1


0


1


0


0


Practice Problem [True/False]

Let $M = (\Sigma, Q, s, A, \delta)$ and $M' = (\Sigma, Q, s, Q \setminus A, \delta)$ be arbitrary DFAs with identical alphabets, states, starting states, and transition functions, but with complementary accepting states. Then $L(M) \cup L(M') = \Sigma^*$.

Practice Problem [True/False]

Let $M = (\Sigma, Q, s, A, \delta)$ and $M' = (\Sigma, Q, s, Q \setminus A, \delta)$ be arbitrary DFAs with identical alphabets, states, starting states, and transition functions, but with complementary accepting states. Then $L(M) \cup L(M') = \Sigma'$.

Closure of Regular languages

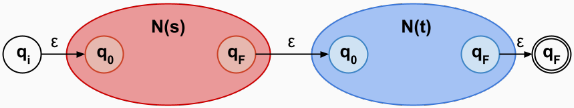
Regular languages are closed under:

-
-
-

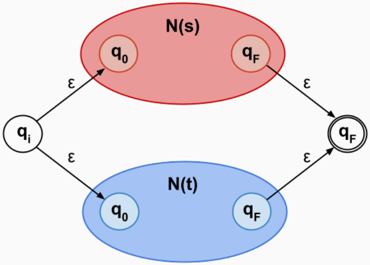
Thompson's algorithm

Given two NFAs s and t:

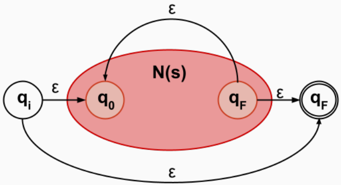
$L = L_s \cap L_t$



$L = L_s \cup L_t$



$L = (L_s)^*$



Example - Closure

Are regular languages closed under intersection $L_1 \cap L_2$?

Practice Problem [True/False]

If $L_1, L_2, \ldots$ are all regular languages, then $L = \bigcup_{i=0}^{\infty} L_i$ is regular.

Fooling Sets

Fooling Sets

Definition
For a language L over Σ a set of strings F (could be infinite) is a fooling set or distinguishing set for L if every two distinct strings $x, y \in F$ are distinguishable.

Fooling Sets

Definition
For a language L over Σ a set of strings F (could be infinite) is a fooling set or distinguishing set for L if every two distinct strings $x, y \in F$ are distinguishable.

Example: $F = \{0^i \mid i \geq 0\}$ is a fooling set for the language $L = \{0^k 1^k \mid k \geq 0\}$.

Fooling Sets

Definition
For a language L over Σ a set of strings F (could be infinite) is a fooling set or distinguishing set for L if every two distinct strings $x, y \in F$ are distinguishable.

Example: $F = \{0^i \mid i \geq 0\}$ is a fooling set for the language $L = \{0^k 1^k \mid k \geq 0\}$.

Theorem
Suppose F is a fooling set for L. If F is finite then there is no DFA M that accepts L with less than $|F|$ states.

Practice Problem [True/False]

For all languages L, if L is regular, then L does not have an infinite fooling set.

Practice Problem [True/False]

The language $\left\{ 0^i 1^j 0^k 1^\ell \;\middle|\; i \geq j \geq k \geq \ell \geq 0 \right\}$ is not regular.

Practice Problem [True/False]

The strings 010 and 101 are distinguishable by the language $L = \{x \in \Sigma^* \mid |x| \text{ is even}\}$.