Pre-lecture brain teaser

Consider the problem of a n-input *AND* function. The input (x) is a string n-digits long with $\Sigma = \{0, 1\}$ and has an output (y) which is the logical *AND* of all the elements of x.

Formulate a **language** that describes the above problem.

CS/ECE-374: Lecture 2 - Regular Languages

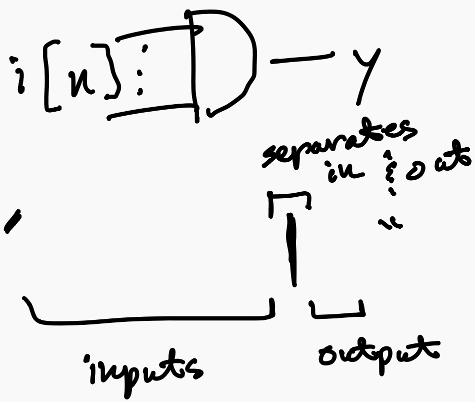
Lecturer: Nickvash Kani
Chat moderator: Samir Khan

January 28, 2021

University of Illinois at Urbana Champaign

Consider the problem of a n-input *AND* function. The input (x) is a string n-digits long with $\Sigma = \{0, 1\}$ and has an output (y) which is the logical *AND* of all the elements of x. $n > 0$

Formulate a **language** that describes the above problem.



$L_{AND} = \{$ "0|0", "1|1",
"00|0", "01|0", "10|0", "11|1",
"000|0", ..., "111|1",
"0000|0", ... $\}$

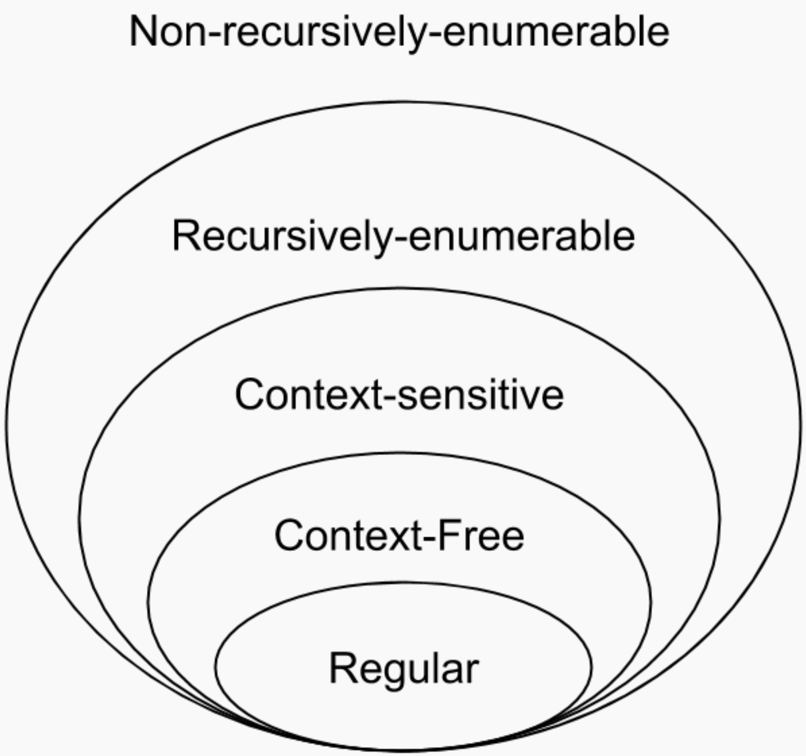
Pre-lecture brain teaser

Consider the problem of a n-input *AND* function. The input (x) is a string n-digits long with $\Sigma = \{0, 1\}$ and has an output (y) which is the logical *AND* of all the elements of x.

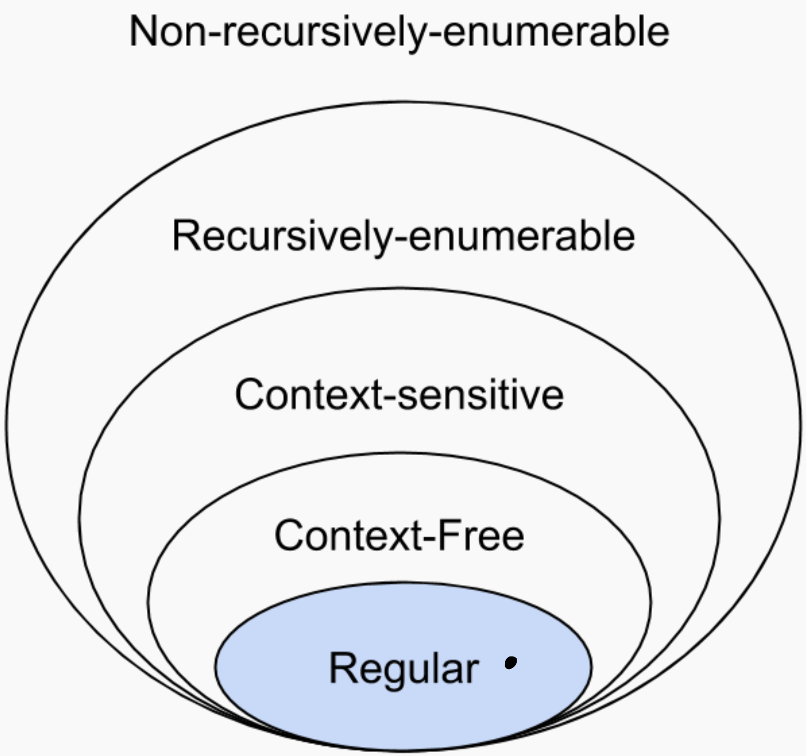
Formulate a **language** that describes the above problem.

This is an example of a regular language which we'll be discussing today.

Chomsky Hierarchy



Chomsky Hierarchy



• L
inf sized
string

Regular Languages

Theorem (Kleene's Theorem)

A language is regular if and only if it can be obtained from finite languages by applying the three operations:

- *Union*
- *Concatenation*
- *Repetition* (*)

a finite number of times.

Regular Languages

A class of simple but useful languages.

The set of regular languages over some alphabet Σ is defined inductively.

Base Case

- $\emptyset$ is a regular language.
- $\{\epsilon\}$ is a regular language.
- $\{a\}$ is a regular language for each $a \in \Sigma$. Interpreting a as string of length 1.

{"ab", "ba"}

Inductive step:

We can build up languages using a few basic operations:

- If L_1, L_2 are regular then $L_1 \cup L_2$ is regular.
- If L_1, L_2 are regular then $L_1 L_2$ is regular.
- If L is regular, then $L^* = \cup_{n \geq 0} L^n$ is regular.
  The $\cdot^*$ operator name is *Kleene star*.
- If L is regular, then so is $\overline{L} = \Sigma^* \setminus L$.

Regular languages are closed under operations of union, concatenation and Kleene star.

Have basic operations to build regular languages.

Important: Any language generated by a finite sequence of such operations is regular.

Lemma
Let $L_1, L_2, \ldots,$ be regular languages over alphabet Σ. Then the language $\cup_{i=1}^{\infty} L_i$ is not necessarily regular.

$L = \bigcup_{i=1}^{\infty} L_i$

Example:

Define $L_i = \{0^i 1^i\}$ * One string in each L_i

$L = \bigcup_{i=1}^{\infty} L_i = \{0^n 1^n \mid n \geq 0\}$ = $\{0^i 1\}^*$ 010101

01
00 11
000 111

not regular

0 , 1 ,
000 111
000000 111

$= 0^* 1^*$

Some simple regular languages

Lemma
If w is a string then $L = \{w\}$ is regular.

Example: $\{aba\}$ or $\{abbabbab\}$. Why?

$L_a = \{\text{"}a\text{"}\}$

$L_b = \{\text{"}b\text{"}\}$

$\{aba\} = L_a L_b L_a$

Some simple regular languages

Lemma
If w is a string then $L = \{w\}$ is regular.

Example: $\{aba\}$ or $\{abbabbab\}$. Why?

Lemma
Every finite language L is regular.

Examples: $L = \{a, abaab, aba\}$. $L = \{w \mid |w| \leq 100\}$. Why?

$\mathbb{R} = \{0, \quad , 1\}$

0.0000000

0.12345 ...

Rapid-fire questions - regular languages

1. $L_1 = \left\{ 0^i \;\middle|\; i = 0, 1, \ldots, \infty \right\}$. The language L_1 is regular. T/F?

$L_0 = \{\text{"}0\text{"}\}$ $\qquad$ $L_1 = L_0^*$

0

00

000

0000

Rapid-fire questions - regular languages

1. $L_1 = \left\{ 0^i \;\middle|\; i = 0, 1, \ldots, \infty \right\}$. The language L_1 is regular. T/F?
2. $L_2 = \left\{ 0^{17i} \;\middle|\; i = 0, 1, \ldots, \infty \right\}$. The language L_2 is regular. T/F? TRUE

$L_0 = \{0\}$ reg

$L_{17 \cdot 0} = \{ L_0 L_0 \cdots L_0 \}$ (17 times) reg

$L_2 = (L_{17 \cdot 0s})^*$ (1 operation) reg

Rapid-fire questions - regular languages

1. $L_1 = \left\{ 0^i \;\middle|\; i = 0, 1, \dots, \infty \right\}$. The language L_1 is regular. T/F?
2. $L_2 = \left\{ 0^{17i} \;\middle|\; i = 0, 1, \dots, \infty \right\}$. The language L_2 is regular. T/F?
3. $L_3 = \left\{ 0^i \;\middle|\; i \text{ is divisible by } 2, 3, \text{or } 5 \right\}$. L_3 is regular. T/F?

$L_0 = \{\text{"}0\text{"}\}$ $\quad L_0 L_0 = \{\text{"}00\text{"}\}$

$L_{i/2} = (L_0 L_0)^*$ ← reg

$L_{i/3} = (L_0 L_0 L_0)^*$ ← reg

$L_{i/5} = (L_0 L_0 L_0 L_0 L_0)^*$ ← reg

$L_3 = L_{i/2} \cup L_{i/3} \cup L_{i/5}$

Rapid-fire questions - regular languages

all binary strings

4. $L_4 = \{w \in \{0,1\}^* \mid w \text{ has at most } \cancel{374}\ 3 \text{ 1s}\}$. L_4 is regular. T/F? TRUE

w is a binary string that has at most 3 1's

$L_0 = \{\text{"0"}\}$ $L_\epsilon = \{\epsilon\}$ $L_{\epsilon 1} = L_\epsilon \cup L_1$

$L_1 = \{\text{"1"}\}$ $L_4 = L_0^* L_{\epsilon 1} L_0^* L_{\epsilon 1} L_0^* L_{\epsilon 1} L_0^*$

reg → $L_{4\_3} = L_0^* L_1 L_0^* L_1 L_0^* L_1 L_0^*$ ← all string with 3 1's

reg → $L_{4\_2} = L_0^* L_1 L_0^* L_1 L_0^*$ $L_{4\_1} = \ldots$ ← all strings with 2 1's

reg → $L_{4\_0} = \ldots$

$0^*\ 1$

reg → $L_4 = L_{4\_0} \cup L_{4\_1} \cup L_{4\_2} \cup L_{4\_3}$

$\Sigma^0 = \epsilon$

$L_0 = \{0\}$
$L_1 = \{1\}$

$\boxed{0^* 1^* 0^*}$

$$\left\{ \begin{array}{l} 010 \\ 0100 \\ 010000 \\ \vdots \end{array} \right. \quad - \quad \left\{ \begin{array}{l} \epsilon \\ 0 \\ 00 \\ 000 \\ \vdots \end{array} \right\} \left\{ \begin{array}{l} 1 \\ 11 \\ 111 \\ \vdots \end{array} \right\} \left\{ \begin{array}{l} 0 \\ 00 \\ 000 \\ \vdots \end{array} \right\}$$

Regular Expressions

$$L_1 L_2 = \{ xy \mid x \in L_1, y \in L_2 \}$$

$$\left\{ \begin{array}{llll} 010 & 0110 & 0010 & 00110 \\ 0100 & 01100 & 00100 & 001100 \\ 01000 & 011000 & 001000 & 0011000 \\ 010000 & & & \\ \vdots & \vdots & \vdots & \vdots \end{array} \right.$$

Regular Expressions

A way to denote regular languages

- simple patterns to describe related strings
- useful in
    - text search (editors, Unix/grep, emacs)
    - compilers: lexical analysis
    - compact way to represent interesting/useful languages
    - dates back to 50's: Stephen Kleene
      who has a star names after him [1].

[1] Kleene, Stephen C.: "Representation of Events in Nerve Nets and Finite Automata". In Shannon, Claude E.; McCarthy, John. Automata Studies, Princeton University Press. pp. 3–42., 1956.

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 $R_1 \cup R_2$
$R_1 R_2$ regular if both are	$\mathbf{r_1} \bullet \mathbf{r_2}$ denotes $R_1 R_2$
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

Notation and Parenthesis

- For a regular expression **r**, $L(\mathbf{r})$ is the language denoted by **r**. Multiple regular expressions can denote the same language!
  Example: $(0 + 1)$ and $(1 + 0)$ denote same language $\{0, 1\}$

Notation and Parenthesis

- For a regular expression $\mathbf{r}$, $L(\mathbf{r})$ is the language denoted by $\mathbf{r}$. Multiple regular expressions can denote the same language!
  Example: $(0+1)$ and $(1+0)$ denote same language $\{0,1\}$
- Two regular expressions $\mathbf{r}_1$ and $\mathbf{r}_2$ are equivalent if $L(\mathbf{r}_1) = L(\mathbf{r}_2)$.

Notation and Parenthesis

- For a regular expression $\mathbf{r}$, $L(\mathbf{r})$ is the language denoted by $\mathbf{r}$. Multiple regular expressions can denote the same language!
  Example: $(0+1)$ and $(1+0)$ denote same language $\{0,1\}$
- Two regular expressions $\mathbf{r}_1$ and $\mathbf{r}_2$ are equivalent if $L(\mathbf{r}_1) = L(\mathbf{r}_2)$.
- Omit parenthesis by adopting precedence order: $*$, concatenate, $+$.
  Example: $r^*s + t = ((r^*)s) + t$

Notation and Parenthesis

- For a regular expression $\mathbf{r}$, $L(\mathbf{r})$ is the language denoted by $\mathbf{r}$. Multiple regular expressions can denote the same language!
  Example: $(0+1)$ and $(1+0)$ denote same language $\{0,1\}$
- Two regular expressions $\mathbf{r}_1$ and $\mathbf{r}_2$ are equivalent if $L(\mathbf{r}_1) = L(\mathbf{r}_2)$.
- Omit parenthesis by adopting precedence order: $*$, concatenate, $+$.
  Example: $r^*s + t = ((r^*)s) + t$
- Omit parenthesis by associativity of each of these operations.
  Example: $rst = (rs)t = r(st)$,
  $r + s + t = r + (s + t) = (r + s) + t$.

Notation and Parenthesis

- For a regular expression $\mathbf{r}$, $L(\mathbf{r})$ is the language denoted by $\mathbf{r}$. Multiple regular expressions can denote the same language!

  Example: $(0+1)$ and $(1+0)$ denote same language $\{0,1\}$
- Two regular expressions $\mathbf{r}_1$ and $\mathbf{r}_2$ are equivalent if $L(\mathbf{r}_1) = L(\mathbf{r}_2)$.
- Omit parenthesis by adopting precedence order: $*$, concatenate, $+$.

  Example: $r^*s + t = ((r^*)s) + t$
- Omit parenthesis by associativity of each of these operations.

  Example: $rst = (rs)t = r(st)$,
  $r + s + t = r + (s + t) = (r + s) + t$.

  $\Sigma^0 = \epsilon$

  no epsilon
- Superscript $+$. For convenience, define $\mathbf{r}^+ = \mathbf{rr}^*$. Hence if $L(\mathbf{r}) = R$ then $L(\mathbf{r}^+) = R^+$.

Notation and Parenthesis

- For a regular expression $\mathbf{r}$, $L(\mathbf{r})$ is the language denoted by $\mathbf{r}$. Multiple regular expressions can denote the same language!
  Example: $(0+1)$ and $(1+0)$ denote same language $\{0,1\}$
- Two regular expressions $\mathbf{r}_1$ and $\mathbf{r}_2$ are equivalent if $L(\mathbf{r}_1) = L(\mathbf{r}_2)$.
- Omit parenthesis by adopting precedence order: $*$, concatenate, $+$.
  Example: $r^*s + t = ((r^*)s) + t$
- Omit parenthesis by associativity of each of these operations.
  Example: $rst = (rs)t = r(st)$,
  $r + s + t = r + (s + t) = (r + s) + t$.
- Superscript $+$. For convenience, define $\mathbf{r}^+ = \mathbf{r}\mathbf{r}^*$. Hence if $L(\mathbf{r}) = R$ then $L(\mathbf{r}^+) = R^+$.

Some examples of regular expressions

Interpreting regular expressions

1. $(0+1)^*$: All binary strings

$L_0 =$ $(L_0 \cup L_1)^*$

$L_1 =$

Interpreting regular expressions

1. $(0+1)^*$: could be ϵ
2. $(0+1)^*001(0+1)^*$:

All strings with 001 as substring

0101 001

Interpreting regular expressions

1. $(0+1)^*$:
2. $(0+1)^*001(0+1)^*$:
3. $0^* + \boxed{(0^*10^*10^*10^*)}^*$:

\# of 1's divisible by 3

$L = \{ 0 \quad 0001000001\ 0010$

$0 + 11111$

Interpreting regular expressions

1. $(0+1)^*$:
2. $(0+1)^*001(0+1)^*$:
3. $0^* + (0^*10^*10^*10^*)^*$:

1 0101 ε

4. $(\epsilon + 1)(01)^*(\epsilon + 0)$:

ε 0101010 0

alternating

ε

1010101 0

1 0

Creating regular expressions

1. All strings that end in 1011? ←

$$(0+1)^* \cdot 1011$$

Creating regular expressions

1. All strings that end in 1011?
2. All strings except 11?

All string 3 characters or greater

$\cup \{00, 01, 10\}$

$\cup \{0, 1\} \cup \{\epsilon\}$

$$\epsilon + 0 + 1 + 00 + 01 + 10 + (0+1)^3(0+1)^*$$

Creating regular expressions

abc is a subsequence of string w
if the letters 'a' 'b' 'c'
appear in the string in that order
but not necessarily consecutively

1. All strings that end in 1011?
2. All strings except 11?
3. All strings that do not contain 000 as a subsequence?

Ex = gabtuc
= gcatub

$$1^* (\epsilon + 0)\ 1^*\ (\epsilon + 0)\ 1^*$$

00: $\epsilon \quad 0 \quad \epsilon \quad 0 \quad \epsilon = 00$

0: $\epsilon \quad \epsilon \quad \epsilon \quad 0 \quad \epsilon = 0$

Creating regular expressions

1. All strings that end in 1011?
2. All strings except 11?
3. All strings that do not contain 000 as a subsequence?
4. All strings that do not contain the substring 10?

All zeros must come before all 1's

$0^* 1^*$

Consider the problem of a n-input *AND* function. The input (x) is a string n-digits long with $\Sigma = \{0, 1\}$ and has an output (y) which is the logical *AND* of all the elements of x.

Formulate the regular expression which describes the above language:

L = { "010" "111" "0010" "0010" "1010" "1111" ....

Case where $y = 1$ $\quad 1^* \, 1^n 1$

Case where $y = 0$ $\quad (0+1)^* 0 (1+0)^* 10$

$$\boxed{1^+ \, 1^n 1 + (0+1)^* 0 (1+0)^* 10}$$

Regular expressions in programming

One last expression....

Bit strings with odd number of 0s and 1s

Bit strings with odd number of 0s and 1s

The regular expression is

$$(00 + 11)^*(01 + 10)\left(00 + 11 + (01 + 10)(00 + 11)^*(01 + 10)\right)^*$$

Bit strings with odd number of 0s and 1s

The regular expression is

$$(00+11)^*(01+10)\left(00+11+(01+10)(00+11)^*(01+10)\right)^*$$

(Solved using techniques to be presented in the following lectures...)