- Urbana
- ZJU
- CSP

- Canvas
- Gradescope
- Pinzza
- Website

Linked

eceeelgo.com

Lectures

→ 0 ✓

→ 1 ~

Labs

→ 0 (X)

→ 1 (Tomorrow)

HWs

→

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.

1010   AND: $1 \wedge 0 \wedge 1 \wedge 0 \longrightarrow 0$

$\boxed{1 \times 2 \mid 2} \longrightarrow M \longrightarrow$ YES

$\boxed{1 \times 2 \mid 4} \longrightarrow M \longrightarrow$ NO

ECE-374-B: Lecture 1 - Regular Languages

Instructor: Abhishek Kumar Umrawal

Jan 18, 2024

University of Illinois at Urbana-Champaign

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.

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.

$\varepsilon | 0$
$\varepsilon | 1$

Convention

$$L_{AND_N} = \left\{ \begin{array}{cccc} 0|0, & 1|1, & & \\ 0 \cdot 0|0, & 0 \cdot 1|0, & 1 \cdot 0|0, & 1 \cdot 1|1 \\ \vdots & \vdots & \vdots & \vdots \\ (0\cdot)^n|0, & (0\cdot)^{n-1}1|0, & \ldots & (1\cdot)^n|1\ldots \end{array} \right\} \quad (1)$$

$\{0, 1, \cdot, 1\}$

Pre-lecture brain teaser

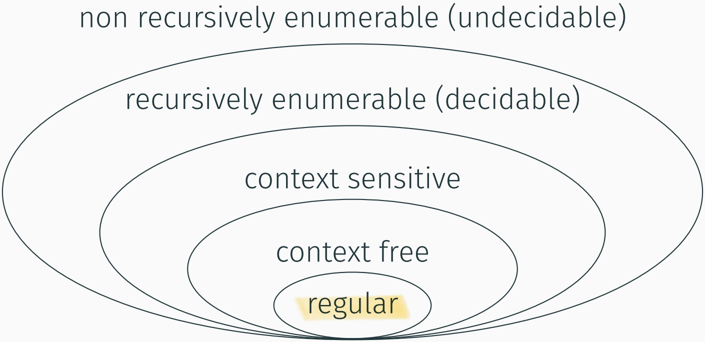
Consider the problem of a n-input AND function. The input (x) is a string n-digits long with $\Sigma = \{0, 1\}$ and has an output (y) which is the logical AND of all the elements of x.

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

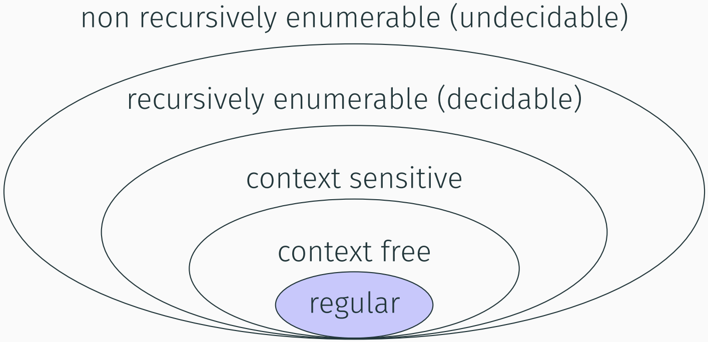
$$L_{AND_N} = \left\{ \begin{matrix} 0|0, & 1|1, & & \\ 0 \cdot 0|0, & 0 \cdot 1|0, & 1 \cdot 0|0, & 1 \cdot 1|1 \\ \vdots & \vdots & \vdots & \vdots \\ (0\cdot)^n|0, & (0\cdot)^{n-1}1|0, & \ldots & (1\cdot)^n|1\ldots \end{matrix} \right\} \tag{1}$$

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

Chomsky Hierarchy



Chomsky Hierarchy



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.

$\underline{\phi}$, $\underline{\{\epsilon\}}$, $\underline{\{a\} \quad a \in \Sigma}$

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

$L^+ \quad L^n$

$\bigcup_{n \geq 0} L^n = \bigcup_{n=0}^{\infty} L^n$

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

Σ^* Eg. $\Sigma = \{0,1\}$, Σ^*

$L = \{00, 11\}$ $\quad \overline{L} = \Sigma^* \setminus \{00, 11\}$

Some simple regular languages

Lemma

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

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

$L = \{aba\}$ Regular ?

By def. $L_a = \{a\}$ is regular!

$L_b = \{b\}$ is regular!

$L = \underline{L_a} \cdot \underline{L_b} \cdot \underline{L_a}$

Hence L is regular!

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?

$L_a = \{a\}$

$L_{abaab} = \{abaab\}$

$L_{aba} = \{aba\}$

Union $\longrightarrow$ L

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 $\boxed{\cup_{i=1}^{\infty} L_i}$ is not necessarily regular.

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.

Note:Kleene star (repetition) is a **single** operation!

Regular Languages - Example

Example: The language $L_{01} = 0^i1^j|$ for all $i, j \geq 0$ is regular:

$$L_{01} = \{\epsilon, 0, 01, 00, 011\}$$

$$010 \notin L_{01}$$

$$\left.\begin{array}{l} L_0 = \{0\} \\ L_1 = \{1\} \end{array}\right\} \rightarrow \text{Regular? } \checkmark$$

$$L_{01} = L_0^* \cdot L_1^*$$

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?

$$\underline{L_1 = \{\epsilon, 0, 00, 000, \ldots\}}$$

$$L_1 = \bigcup_{i \geq 0} \{0\}^i = \{0\}^*$$

$$L^* = \bigcup_{n \geq 0} L^n$$

$$\boxed{\epsilon \notin \{0\}} \checkmark$$

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?

$$L_2 = \{ \underbrace{0 \cdot 0 \cdot \; \cdots \; \cdot 0}_{17 \text{ times}} \}^*$$

$$L_2 = \left(L_0^{17} \right)^*$$

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?
3. $L_3 = \left\{ 0^i \,\middle|\, i \text{ is divisible by } 2, 3, \text{or } 5 \right\}$. L_3 is regular. T/F?

$$L_3 = \{ \epsilon, 00, 0000, \ldots$$
$$000, 000000, \ldots$$
$$00000, \ldots \}$$

$L_{i/2} := (L_0^2)^*$ $\qquad$ $L_{i/5} :=$ DIY

$L_{i/3} :=$ DIY

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?
3. $L_3 = \left\{ 0^i \;\middle|\; i \text{ is divisible by } 2, 3, \text{or } 5 \right\}$. L_3 is regular. T/F?
4. $L_4 = \{ w \in \{0, 1\}^* \mid w \text{ has at most 2 1s} \}$. L_4 is regular. T/F?

(DIY)

Regular Expressions

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].

$\{0\}$ $\qquad$ 0

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_1R_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 $\mathbf{r}$, $L(\mathbf{r})$ is the language denoted by $\mathbf{r}$. Multiple regular expressions can denote the same language!
  Example: $(0+1)$ and $(1+0)$ denotes same language $\{0,1\}$

$r = 0 + 1$ $\qquad$ $r_1 = 1 + 0$

$L(r) = \{0, 1\}$ $\qquad$ $L(r_1) = \{0, 1\}$

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 $(1+0)$ denotes 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 $(1+0)$ denotes 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: $*, \cdot, +$.
  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 $(1+0)$ denotes 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: $*, \cdot, +$.
  Example: $r^*s + t = ((r^*)s) + t$
- Omit parenthesis by associativity of each operation.
  Example: $rst = (rs)t = r(st)$,
  $r + s + t = r + (s + t) = (r + s) + t$.

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 $(1+0)$ denotes 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: $*, \cdot, +$.
  Example: $r^*s + t = ((r^*)s) + t$
- Omit parenthesis by associativity of each operation.
  Example: $rst = (rs)t = r(st)$,
  $r + s + t = r + (s + t) = (r + s) + t$.
- Superscript $+$. For convenience, define $\boxed{\mathbf{r}^+ = \mathbf{r}\mathbf{r}^*.}$ Hence if $L(\mathbf{r}) = R$ then $L(\mathbf{r}^+) = R^+$.

$r^+ = r r^*$

$$\boxed{r^* s \ := \ (L(r))^* \cdot (L(s))}$$

L^* vs L^+

$L^* = L^+ \cup \{\epsilon\}$

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 $(1 + 0)$ denotes 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: $*, \cdot, +$.

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

  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 $L(\mathbf{r}^+) = R^+$.
- Other notation: $r + s$, $r \cup s$, $r|s$ all denote union. rs is sometimes written as $r \bullet s$.

Some examples of regular expressions

Creating regular expressions

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

1. All strings that end in 1011?

1011

01011

11011

$\boxed{}\ 1011$

$$r = (1+0)^* 1011$$

Creating regular expressions

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

$(1+0)^* - 11$ ✓ $\quad (1+0)^* \setminus 11$

$$\varepsilon + 0 + 1 + 00 + 10 + 01 + \cancel{11}$$

$$+ \; (0+1)(0+1)(0+1)^+$$

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?

$\star$ All strings with # of zeros ≤ 2.

$w_1 = 0100101$

000 is a subseq of w_1! (✓)

$w_2 = 010$

000 ——— w_2! (✗)

$\varepsilon + 01^* + 001^* + \cdots \cdots\to$

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

Something along this line ..

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?

110   1 0^*

1^*

$*$ $0^* 1^*$ $0^* 1^*$ ✓

Interpreting regular expressions

1. $(0+1)^*$:

Interpreting regular expressions

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

Interpreting regular expressions

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

Interpreting regular expressions

1. $(0+1)^*$:
2. $(0+1)^*001(0+1)^*$:
3. $0^* + (0^*10^*10^*10^*)^*$:
4. $(\epsilon+1)(01)^*(\epsilon+0)$:

Tying everything together

Consider the problem of a n-input AND function. The input (x) is a string n-digits long with an input alphabet $\Sigma_i = \{0, 1\}$ and has an output (y) which is the logical AND of all the elements of x. We knwo the language used to describe it is:

$$L_{AND_N} = \left\{ \begin{array}{cccc} 0 \cdot |0, & 1 \cdot |1, & & \\ 0 \cdot 0 \cdot |0, & 0 \cdot 1 \cdot |0, & 1 \cdot 0 \cdot |0, & 1 \cdot 1 \cdot |1 \\ \vdots & \vdots & \vdots & \vdots \\ (0 \cdot)^n|0, & (0 \cdot)^{n-1}1|0, & \ldots & (1 \cdot)^n|1 \ldots \end{array} \right\}$$

Formulate the regular expression which describes the above language:

Tying everything together

Consider the problem of a n-input AND function. The input (x) is a string n-digits long with an input alphabet $\Sigma_i = \{0, 1\}$ and has an output (y) which is the logical AND of all the elements of x. We knwo the language used to describe it is:

$$L_{AND_N} = \begin{Bmatrix} 0\cdot|0, & 1\cdot|1, & & \\ 0\cdot 0\cdot|0, & 0\cdot 1\cdot|0, & 1\cdot 0\cdot|0, & 1\cdot 1\cdot|1 \\ \vdots & \vdots & \vdots & \vdots \\ (0\cdot)^n|0, & (0\cdot)^{n-1}1|0, & \dots & (1\cdot)^n|1\dots \end{Bmatrix}$$

Formulate the regular expression which describes the above language: $\Sigma = \{0, 1, `\cdot', `|'\}$

$$r_{AND_N} = \underbrace{(\text{“}0\cdot\text{”} + \text{“}1\cdot\text{”})^* \text{“}0\cdot\text{”} (\text{“}0\cdot\text{”} + \text{“}1\cdot\text{”})^* \text{“}|0\text{”}}_{\text{all output 0 instances}} + \overbrace{(\text{“}1\cdot\text{”})^* \text{“}|1\text{”}}^{\text{all output 1 instances}}$$

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

$$(00 + 11)^*(01 + 10)$$
$$\Big(00 + 11 + (01 + 10)(00 + 11)^*(01 + 10)\Big)^*$$

Bit strings with odd number of 0s and 1s

The regular expression is

$$(00+11)^*(01+10)\Big(00+11+(01+10)(00+11)^*(01+10)\Big)^*$$

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