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.

ECE-374-B: Lecture 2 - Regular Languages

Lecturer: Nickvash Kani

January 19, 2023

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.

$$f(x) = x_0 \cdot x_1 \cdot x_2 \cdot x_3 \cdots = y$$

Pre-lecture brain teaser

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

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

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

Ø|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.

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

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

Strings

Alphabet

An alphabet is a **finite** set of symbols.

Examples of alphabets:

- $\Sigma = \{0, 1\}$,
- $\Sigma = \{a, b, c, \ldots, z\}$,
- ASCII.
- UTF8.
- $\Sigma = \{\langle\text{moveforward}\rangle, \langle\text{moveback}\rangle, \langle\text{moveleft}\rangle, \langle\text{moveright}\rangle\}$

String Definition

$\emptyset$

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. $x \cdot y \equiv xy$ is the concatenation of two strings
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. $\Sigma = \{0, 1\}$ $\Sigma^2 = \{00, 01, 10, 11\}$
5. Σ^* set of all strings of all lengths including empty string.

$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \ldots$

Question: $\{'a', 'c'\}^* = \{\epsilon, a, c, aa, ac, ca, cc, aaa, aac \ldots$

$\Sigma^* = \bigcup_{n=0}^{\infty} \Sigma^n$

'hi'

"hi"

Emptiness

- ϵ is a string containing no symbols. It is not a set
- $\{\epsilon\}$ is a set containing one string: the empty string. It is a set, not a string.
- $\emptyset$ is the empty set. It contains no strings.

Question: What is $\{\emptyset\}$ set containing an empty set

Concatenation and properties

- If x and y are strings then xy denotes their concatenation.
- Concatenation defined recursively : recursive definition
    - $xy = y$ if $x = \epsilon$
    - $xy = a(wy)$ if $x = aw$
- xy sometimes written as $x \bullet y$.
- concatenation is associative: $(uv)w = u(vw)$ hence write $uvw \equiv (uv)w = u(vw)$
- **not** commutative: uv not necessarily equal to vu $u = 1$ $v = 2$ $12 \neq 21$
- The identity element is the empty string ϵ:

$$\epsilon u = u\epsilon = u.$$

Substrings, prefixes, Suffixes

Definition
v is substring of $w \iff$ there exist strings x, y such that $w = xvy$.

- If $x = \epsilon$ then v is a prefix of w
- If $y = \epsilon$ then v is a suffix of w

Subsequence

A subsequence of a string $w[1...n]$ is either a subsequence of $w[2...n]$ or $w[1]$ followed by a subsequence of $w[2...n]$.

Example
kapa is a sub-sequence of *knapsack*

kapa

knapsack

knap

kaps

Subsequence

A subsequence of a string $w[1...n]$ is either a subsequence of $w[2...n]$ or $w[1]$ followed by a subsequence of $w[2...n]$.

Example
kapa is a sub-sequence of *knapsack*

Question: How many sub-sequences are there in a string $|w| = 5$?

2^5

— — — — —

a a a a a

ϵ a aa aaa aaaa

aaaaa

String exponent

Definition
If w is a string then w^n is defined inductively as follows:

$w^n = \epsilon$ if $n = 0$

$w^n = ww^{n-1}$ if $n > 0$

Question: $(blah)^3 =$. blah blah blah

Rapid-fire questions -strings

Answer the following questions taking $\Sigma = \{0, 1\}$.

1. What is Σ^0? $\{\epsilon\}$
2. How many elements are there in Σ^n? 2^n
3. If $|u| = 2$ and $|v| = 3$ then what is $|u \bullet v|$? 5
4. Let u be an arbitrary string in Σ^*. What is ϵu? What is $u\epsilon$?

u

Languages

Languages

Definition
A language L is a set of strings over Σ. In other words $L \subseteq \Sigma^*$.

Languages

Definition
A language L is a set of strings over Σ. In other words $L \subseteq \Sigma^*$.

Standard set operations apply to languages.

- For languages A, B the concatenation of A, B is $AB = \{xy \mid x \in A, y \in B\}$.
- For languages A, B, their union is $A \cup B$, intersection is $A \cap B$, and difference is $A \setminus B$ (also written as $A - B$).
- For language $A \subseteq \Sigma^*$ the complement of A is $\bar{A} = \Sigma^* \setminus A$.

Definition
Given two sets X and Y of strings (over some common alphabet Σ) the concatenation of X and Y is

$$XY = \{xy \mid x \in X, y \in Y\} \tag{2}$$

Question: $X = \{fido, rover, spot\}$, $Y = \{fluffy, tabby\} \implies$
$XY = .$ {fidofluffy
fidotabby
roverfluffy
...
}

Σ^* and languages

$|\Sigma^n| = 2^n$ elements

$\Sigma \cdot \Sigma^{n-1}$

Definition

1. Σ^n is the set of all strings of length n. Defined inductively:
   $\Sigma^n = \{\epsilon\}$ if $n = 0$
   $\Sigma^n = \Sigma\Sigma^{n-1}$ if $n > 0$
2. $\Sigma^* = \cup_{n \geq 0} \Sigma^n$ is the set of all finite length strings
3. $\Sigma^+ = \cup_{n \geq 1} \Sigma^n$ is the set of non-empty strings.

Definition
A language L is a set of strings over Σ. In other words $L \subseteq \Sigma^*$.

Question: Does Σ^* have strings of infinite length?

Problem
Consider languages over $\Sigma = \{0, 1\}$.

1. *What is* $\emptyset^0$? $\{\epsilon\}$
2. *If* $|L| = 2$, *then what is* $|L^4|$? $2^4 = 16$
3. *What is* $\emptyset^*$, $\{\epsilon\}^*$, ϵ^*? $\{\epsilon\}$
4. *For what L is* L^* *finite?* $\{\}$ $\{\epsilon\}$
5. *What is* $\emptyset^+$? $\{\}$
6. *What is* $\{\epsilon\}^+$, ϵ^+? $\{\epsilon\}$

$\emptyset^* = \cup \emptyset^0 \cup \emptyset^1 \cup \emptyset^2 \ldots$

ϵ

$\emptyset^+ = \cup \emptyset^1 \cup \emptyset^2 \cup$

$\epsilon^+ = \epsilon^1 \cup \epsilon^2 \cup \epsilon^3 \cup \epsilon^4$

$\{\epsilon\}\{\epsilon\}\{\epsilon\} = \{\epsilon\}^3$

$= \epsilon^1$

$\{\epsilon\}^+ = \{\epsilon\}^1 \cup \{\epsilon\}^2 \cup \{\epsilon\}^3$

$\{\epsilon, \epsilon\epsilon, \epsilon\epsilon\epsilon\}$

$\epsilon \cup \epsilon \cdot \epsilon \cup \epsilon \cdot \epsilon \cdot \epsilon \cup$

Terminology Review

Let's review what we learned.

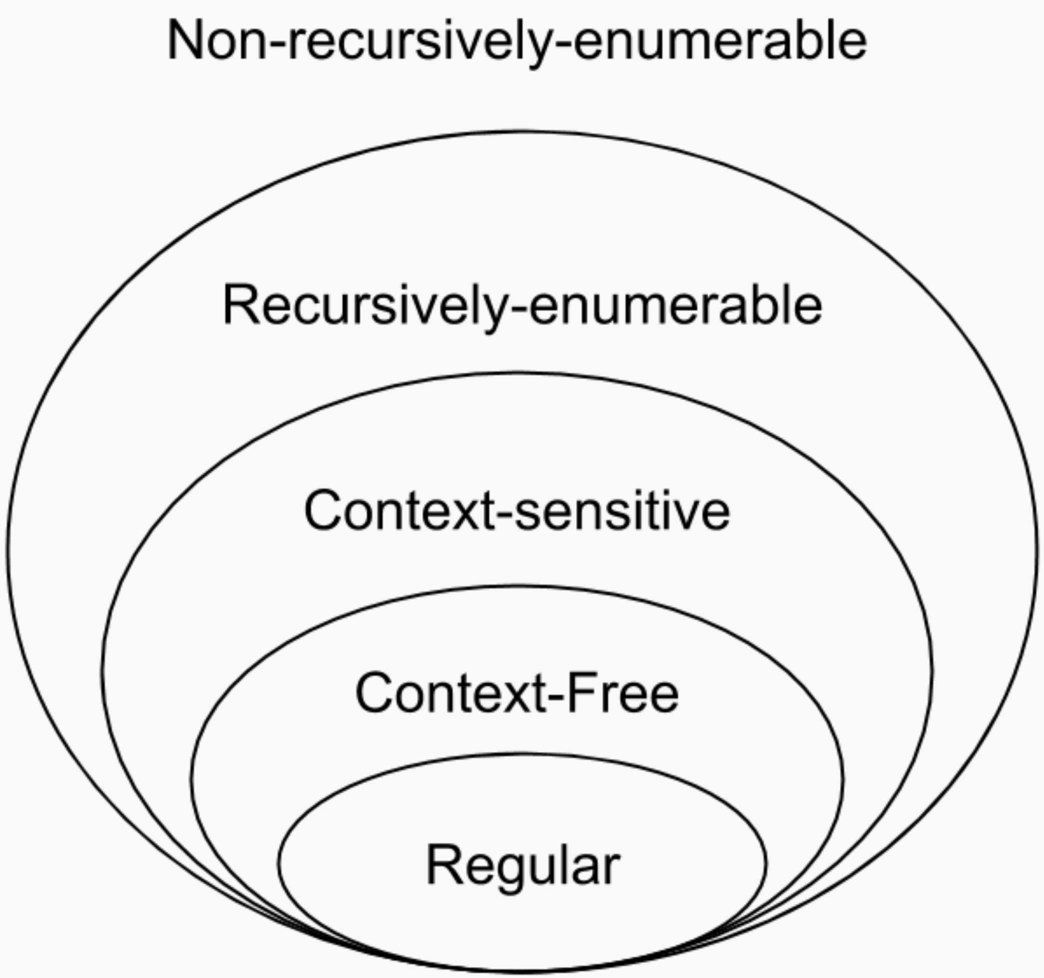
- A character(a, b, c, x) is a unit of information represented by a symbol: (letters, digits, whitespace)
- A alphabet(Σ) is a set of characters
- A string(w) is a sequence of characters
- A language(A, B, C, L) is a set of strings

Terminology Review

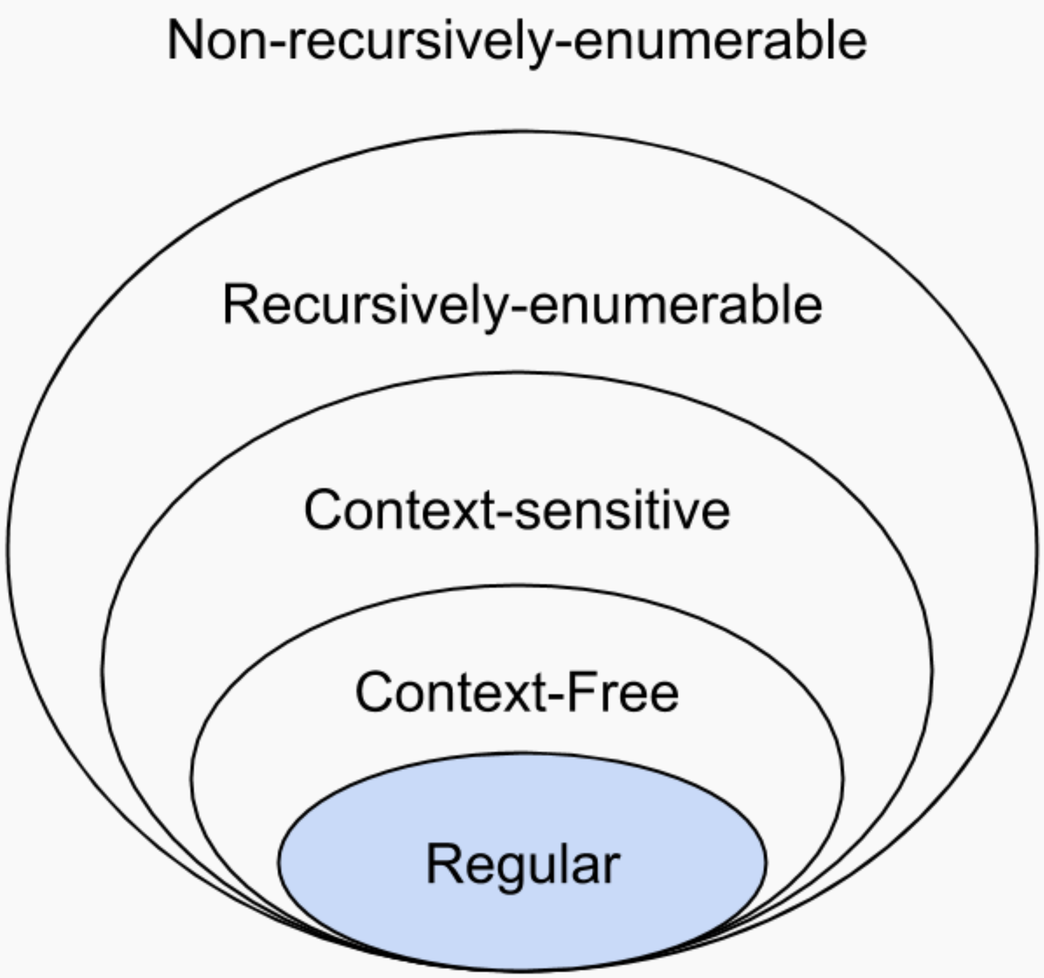
Let's review what we learned.

- A character(a, b, c, x) is a unit of information represented by a symbol: (letters, digits, whitespace)
- A alphabet(Σ) is a set of characters
- A string(w) is a sequence of characters
- A language(A, B, C, L) is a set of strings
- A grammar(G) is a set of rules that defines the strings that belong to a language

Chomsky Hierarchy



Chomsky Hierarchy



Regular Languages

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.

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.

Some simple regular languages

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

$\Sigma = \{a, b\}$

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

Base: $L_A = \{a\}$

$L_B = \{b\}$

$L_{aba} = L_A \cdot L_B \cdot L_A$

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?

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.

Regular Languages

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!

L_1^* is regular

$$L_1^* = \bigcup_{n=0}^{\infty} L_1^n$$

Regular Languages - Example

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

All strings that have a run of 0's concatenated with a run of 1's

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

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

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

$$L_{01}^* = (L_0^* L_1^*)^*$$

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?

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?

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?

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. $L_4 = \{w \in \{0, 1\}^* \mid w \text{ has at most } 2 \text{ 1s}\}$. L_4 is regular. T/F?

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].

Inductive Definition

A regular expression $\mathbf{r}$ over an alphabet $\boldsymbol{\Sigma}$ 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 $\mathbf{r}$, $L(\mathbf{r})$ is the language denoted by $\mathbf{r}$. Multiple regular expressions can denote the same language!
  Example: $(0+1)$ and $(1+0)$ denotes same language $\{0,1\}$

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 **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: $*$,
  $(\cdot)$ 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 $(1 + 0)$ 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: $*$, 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 $(1+0)$ 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: $*$, 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 $L(\mathbf{r}^+) = R^+$.

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: $*$, 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

Interpreting regular expressions

1. $(0+1)^*$:
2. $(0+1)^*001(0+1)^*$: All binary with 001 substring

Interpreting regular expressions

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

$r_1 = 0^* 1 0^* 1 0^* 1 0^*$ All strings with 3 1's

$r_1^* = \ldots$ All strings with a number of 1's divisible by 3

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

- All strings with alternating 0's & 1's
- No two consecutive 0's and no two consecutive 1's

Creating regular expressions

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

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?

$\cancel{(0+1)^* - 11}$

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

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?

All strings with at most two 0's

$$1^* 0 1^* 0 1^* + 1^* 0 1^* + 1^*$$

$\{\epsilon, 1, 11, \ldots\}$

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

$\epsilon \cdot \epsilon \cdot \epsilon \cdot \epsilon \cdot \epsilon \cdots \epsilon$

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?

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:

$$L_{AND_N} = \left\{ \begin{array}{cccc} 0|0, & 1|1, & & \\ 0 \cdot 0|0, & 0 \cdot 1|0, & 1 \cdot 0|0, & 1 \cdot 1|1 \\ \vdots & \vdots & \vdots & \vdots \\ (0\cdot)^n|0, & (0\cdot)^{n-1}1|0, & \dots & (1\cdot)^n|1\dots \end{array} \right\} \tag{3}$$

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:

1 = 1 00 11

$$L_{AND_N} = \left\{ \begin{matrix} 0|0, & 1|1, & & \\ 0 \cdot 0|0, & 0 \cdot 1|0, & 1 \cdot 0|0, & 1 \cdot 1|1 \\ \vdots & \vdots & \vdots & \vdots \\ (0\cdot)^n|0, & (0\cdot)^{n-1}1|0, & \ldots & (1\cdot)^n|1 \ldots \end{matrix} \right\} \tag{3}$$

0×1 = 0

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

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

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

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

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