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 1 - Regular Languages

Lecturer: Nickvash Kani

August 24, 2023

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.

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

$\cdot = 1$

$0\cdot = 0$

$1\cdot = 1$

$\ldots$

$AND_N = \{$ "$0\cdot = 0$", "$1\cdot = 1$", "$0\cdot 0 = 0$", "$0\cdot 1 = 0$", "$1\cdot 0 = 0$", "$1\cdot 1 = 1$", "$1111|1$", $\ldots$ $\}$

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.

$\in L_{AND_N}$

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

"$0 \cdot 0 | 1$" $\notin L_{AND_N}$

Pre-lecture brain teaser

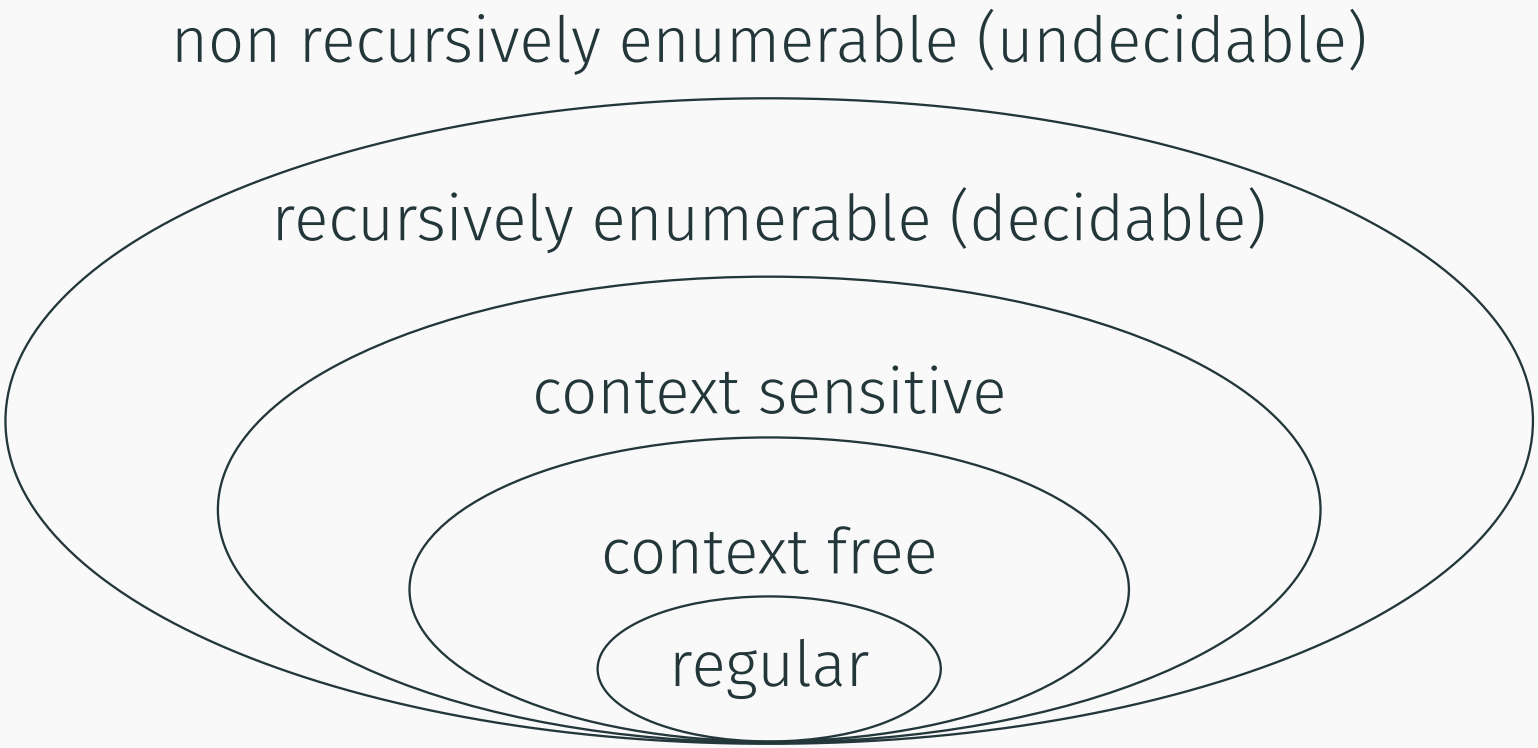
Consider the problem of a n-input AND function. The input (x) is a string n-digits long with $\Sigma = \{0, 1\}$ and has an output (y) which is the logical AND of all the elements of x.

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

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

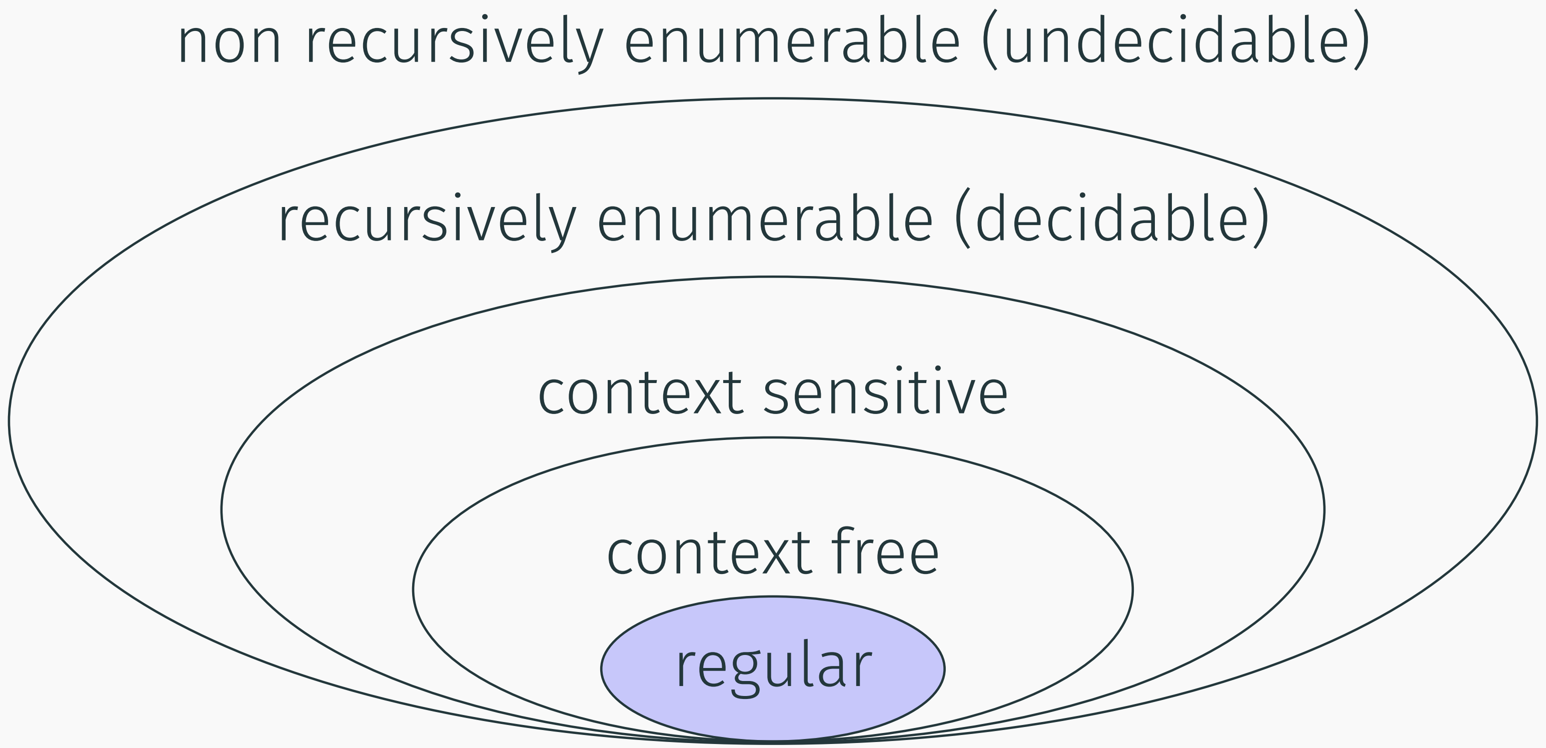
This is an example of a regular language which we'll be discussing today.

Chomsky Hierarchy



$$L_{w_0} + L_{w_1} + L_{w_2} + \ldots L_{w_\infty} = L : \left| \left\{ \begin{matrix} w_0 \\ w_1 \\ w_2 \\ \vdots \end{matrix} \right\} \right| = \infty$$

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 $\overline{L} = \Sigma^* \setminus L$.

$(0+1)^*$

$(L_0 \cup L_1)^*$

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

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

$\bigcup_{n>0}^{5} L^n \cup \bigcup_{n>0}^{6} L^n$

Some simple regular languages

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

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

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

Base: $\emptyset$

$\{\epsilon\}$ $\{a\}$ $\{b\}$ $\longrightarrow \{aba\}$

L_a L_b

$L_{aba} = L_a \cdot L_b \cdot L_a$

$L_{abbabbab} = L_a \cdot L_b \cdot L_b \cdot L_a \cdot L_b \cdot L_b \cdot L_a \cdot L_b$

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: $L_3 = \underbrace{\{a, abaab, aba\}}$. $L = \{w \mid |w| \leq 100\}$. Why?

$L_a \cup L_{abaab} \cup L_{aba} = L_3$

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.

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:

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

$L_\epsilon = \{\epsilon\}$

$L_0 = \{0\}$

$L_1 = \{1\}$

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

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?

$L_1 = L_0^* = \bigcup_{n>0}^{\infty} L_0^n$

$= L_0^0 \cup L_0^1 \cup L_0^2$

$L^+ = L^* \cdot L^* L^* = \{\varepsilon\} \cup \{0\} \cup \{0\,0\} \cup$

$= \{\epsilon, 0, 00, \ldots\}$

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?

$$L_2 = \Big(\underbrace{L_0 \cdot L_0 \cdot \ldots \ldots \cdot L_0}_{17 \text{ times}} \Big)^*$$

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?

$$L_{20's} = (L_0 \cdot L_0)^*$$

$$L_{30's} = (L_0 \cdot L_0 \cdot L_0)^*$$

$$L_{50's} = (L_0 \cdot L_0 \cdot L_0 \cdot L_0 \cdot L_0)^*$$

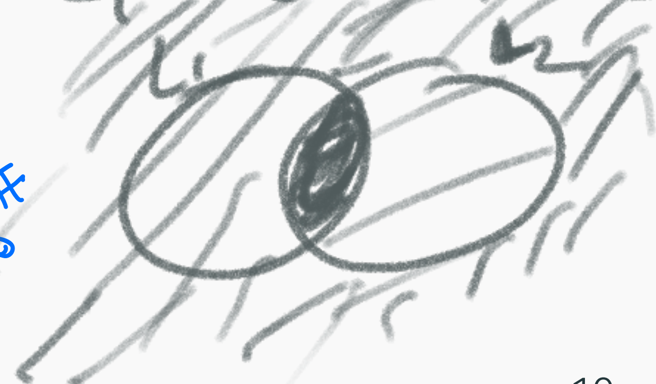
$$L_3 = L_{20's} \cup L_{30's} \cup L_{50's}$$

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

$$L_4 = \begin{cases} L_{0 1's} = L_0^* \\ L_{1 1's} = L_0^* L_1 L_0^* \\ L_{2 1's} = L_0^* L_1 L_0^* L_1 L_0^* \end{cases}$$

$$L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$$



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 **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

$L_1 = \{0^i \mid i \geq 0\}$

Regular Languages	Regular Expressions
$\emptyset$ regular	$\emptyset$ denotes $\emptyset$
$\{\epsilon\}$ regular	ϵ denotes $\{\epsilon\}$
$\{a\}$ regular for $a \in \Sigma$	**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^*

$r = 0^*$

$L(r) = L_1$

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

$R_1 = L_a \cup L_b = \{a, b\} = a + b$

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 $\{0,1\}$

$\{0\} \cup \{0\} = \{0\}$

$0 + 0 + 0 + 0 + \ldots + 0 =$

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 $\mathbf{r}$, $L(\mathbf{r})$ 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 $L(\mathbf{r}^+) = R^+$.

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

1. All strings that end in 1011?

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

Creating regular expressions

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

$$(0+1)^* - 11$$

$$\cancel{(0+1)^* 00 + (0+1)^* 01 + (0+1)^* 10}$$

$$\underbrace{\varepsilon + 0 + 1 + 00 + 01 + 10} + \cancel{(0+1)(\varepsilon + 1)}$$

$$\underbrace{+ (0+1)(0+1)(0+1)^+}$$

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

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?

$\{\varepsilon, 1, 11, 11\cdots\}$

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

$1^* + 1^*01^* + 1^*01^*01^*$

$(1)^*(0)(1^*)(0)(1^*)$

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?
4. All strings that do not contain the substring 10?

Interpreting regular expressions

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

Interpreting regular expressions

1. $(0+1)^*$:
2. $(0+1)^*001(0+1)^*$: All string that have 001 as a substring

Interpreting regular expressions

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

All string whose $\#_1$'s is divisible by 3

Interpreting regular expressions

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

Alternating

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:

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

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...)