Strings, Languages, and Regular expressions
Definitions for strings

- $\Sigma = \text{finite alphabet of symbols}$
  - $\Sigma = \{0,1\}$, or $\Sigma=\{a,b,c,...,z\}$, or $\Sigma=$all ascii characters
- string or word = finite sequence of symbols of $\Sigma$
- length of a string $w$ is denoted $|w|$.  $|\text{cat}|=3$
- the empty string is denoted “$\varepsilon$”.  $|\varepsilon| = 0$.

Conventions
- $a, b, c, ...$ denote strings of length 1; elements of $\Sigma$
- $w, x, y, z, ...$ denote strings of length 0 or more
- $A, B, C,...$ denote sets of strings
Much ado about nothing

• $\varepsilon$ is a *string* containing no symbols. It is not a set.
• $\{\varepsilon\}$ is a *set* containing one string: the empty string $\varepsilon$. It is a *set*, not a string.
• $\emptyset$ is the *empty set*. It contains no strings.
• $\{\emptyset\}$ is a *set* containing one element, which itself is a set with no elements.
**Concatenation & its properties**

- If $x$ and $y$ are strings, $xy$ denotes the concatenation.
- Associative: $(uv)w = u(vw)$ and we write $uvw$.
- NOT commutative: $ab \neq ba$.
- Identity element $\varepsilon$: $\varepsilon w = w\varepsilon = w$.
- Length (can be defined inductively):
  - $|\varepsilon| = 0$
  - $|a| = 1$
  - $|au| = 1 + |u|$
Substrings, prefix, suffix, exponents

• \( v \) is a *substring* of \( w \) iff there exist strings \( x, y \), such that \( w = xvy \).
  - If \( x = \varepsilon \) then \( v \) is a *prefix* of \( w \).
  - If \( y = \varepsilon \) then \( v \) is a *suffix* of \( w \).

• If \( w \) is a string, then \( w^i \) is defined inductively by:
  \[ w^i = \varepsilon \text{ if } i = 0 \]
  \[ w^i = ww^{i-1} \text{ if } i > 0. \]

  e.g. \((\text{blah})^4 = \text{blahblahblahblah}\)
Set Concatenation

• If $X$ and $Y$ are sets of strings, then
  
  $XY = \{xy \mid x \text{ in } X \text{ and } y \text{ in } Y\}$

  e.g. $X = \{\text{fido, rover, spot}\}$, $Y = \{\text{fluffy, tabby}\}$
  
  then $XY = \{\text{fidofluffy, fidotabby, roverfluffy, ...}\}$
\( \Sigma^n, \Sigma^*, \text{and} \Sigma^+ \)

- \( \Sigma^n \) defined as all strings over \( \Sigma \) of length \( n \) inductively:
  \[ \Sigma^0 = \{ \varepsilon \} \]
  \[ \Sigma^n = \Sigma \Sigma^{n-1} \text{ if } n > 0 \]

- \( \Sigma^* \) is the set of all finite length strings:
  \[ \Sigma^* = \bigcup_{n \geq 0} \Sigma^n \]

- \( \Sigma^+ \) is the set of all nonempty finite length strings:
  \[ \Sigma^+ = \bigcup_{n \geq 1} \Sigma^n = \Sigma \Sigma^* \]
\[ \Sigma^n, \Sigma^*, \text{ and } \Sigma^+ \]

Examples

- \( \Sigma = \{0, 1\} \). Then \( \Sigma^2 = \{00, 01, 10, 11\} \). \( \Sigma^0 = \{\varepsilon\} \)
- \( \Sigma = \{a, b, c, \ldots, z, A, \ldots, Z, _, -, +, \ldots \ <\text{other symbols}\} \).
  - \( \bigcup_{n \leq 100} \Sigma^n \) contains all English words (and more)
  - \( \Sigma^* \) contains all books sold by Amazon (and more)
- \( \Sigma = \emptyset \). Then \( \Sigma^1 = \Sigma^2 = \ldots = \Sigma^{100} = \emptyset \)
  \[ \Sigma^0 = \{\varepsilon\} \]
\[ \Sigma^* = \bigcup_{n \geq 0} \Sigma^n \]

- What is the cardinality of \( \Sigma^n \)?
  \[ |\Sigma^n| = |\Sigma|^n \]
- What is the cardinality of \( \Sigma^* \)?
  \[ |\Sigma^*| = \aleph_0 = |\mathbb{N}| \quad \text{(provided that \( \Sigma \) is nonempty)} \]
- What is the length of the longest element of \( \Sigma^* \)?
  there is no longest element
- Are there any infinitely long strings in \( \Sigma^* \)?
  \text{NO!} \quad \Sigma^* \text{ has strings of arbitrary size, but no single unbounded (infinite) string}
Canonical Order

• Enumerate $\Sigma^*$ in order of increasing length strings and for strings of same length, in dictionary order

e.g. $\{0,1\}^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, 010, \ldots\}$

$\{a,b\}^* = \{\varepsilon, a, b, aa, ab, ba, bb, aaa, aab, aba, \ldots\}$
Inductive Definitions

• Often strings and functions on strings are defined inductively.

• Example: $w^R$, the reverse of word $w$ is defined:
  - if $|w| = 0$, then $w = \varepsilon$, and $w^R = \varepsilon$
  - if $|w| > 0$, then $w = au$ for some $a$ in $\Sigma$ and $u$ in $\Sigma^*$
    with $|u| < |w|$
  and then $w^R = u^R a$

(abc)$^R = (bc)^R a = (c^R b)a = ((c\varepsilon)^R b)a = ((\varepsilon^R c)b)a = cba$
Inductive proofs follow inductive defs

Theorem: For any strings $u$ and $v$, $(uv)^R = v^R u^R$
  
  e.g. $(dogcat)^R = (cat)^R (dog)^R = tacgod$

Proof: by induction.

On what??

$|uv| = |u| + |v|$ ?

$|u|$ ?

$|v|$ ?

$|u|$ and in induction, do an *inner induction* on $|v|$ ?
Induction on $|u|$ 

Proof: by induction on $|u|$ is most natural

Base case: If $|u| = 0$, then $u = \varepsilon$, and for any $v$,

$$(uv)^R = (\varepsilon v)^R = v^R \varepsilon = v^R \varepsilon^R = v^R u^R$$
**Inductive Step**

- Assume for any $u$ of length $< n$ that:
  
  for all $v$, $(uv)^R = v^R u^R$

- Let $u$ be an arbitrary string of length $n$.
  
  Then $u = ay$ for some $a$ in $\Sigma$ and $|y| < n$

Then

\[
(\overline{uv})^R = (\overline{(ay)v})^R \quad \text{because } u = ay
\]

\[
= (a(yv))^R \quad \text{because concatenation is associative}
\]

\[
= (yv)^R a \quad \text{by inductive definition of reverse}
\]

\[
= (v^R y^R) a \quad \text{applying inductive hypothesis } (|y| < n)
\]

\[
= v^R (y^R a) \quad \text{because concatenation is associative}
\]

\[
= v^R (ay)^R \quad \text{by inductive definition of reverse}
\]

\[
= v^R u^R \quad \text{because } u = ay
\]
Induction on $|v|$

- Base cases need $|v| = 0$ or $1$.
- Assume for any $v$ of length $< n$ that:
  
  for all $v$, $(uv)^R = v^R u^R$

- Let $v$ be an arbitrary string of length $n > 1$.

Then $v = ax$ for some $a$ in $\Sigma$ and $|x| < n$

Then 

$(uv)^R = (u(ax))^R$ because $v = ax$

$= ((ua)x)^R$ because concatenation is associative

$= x^R (ua)^R$ applying inductive hypothesis ($|x| < n$)

$= x^R (a^R u^R)$ applying inductive hypothesis ($|a| < n$)

$= x^R (au^R)$ ($a = a^R$ via definition of reverse)

$= (x^R a) u^R$ because concatenation is associative

$= (ax)^R u^R$ by inductive definition of reverse

$= v^R u^R$ because $v = ax$
Languages

• If $\Sigma$ is a (finite) alphabet, then a language is any subset of $\Sigma^*$ [often, $\Sigma$ is clear from context]

• Thus, a language is just a set of strings (words)

Examples
- $\{\varepsilon\}$
- $\{w : |w| > 5\}$
- $\{w : w$ is a syntactically correct Python program$\}$
- $\{w : w$ is the text of a book in the Library of Congress$\}$
- $\emptyset$
• The **complement** of a language $L$ is $\overline{L} = \Sigma^* - L$
  (where $A - B$ is set subtraction)

• $L^n$, $L^*$, and $L^+$ defined as were $\Sigma^n$, $\Sigma^*$, and $\Sigma^+$
  note that a word in $L^n$ is the
  concatenation of *n possibly different* words in $L$.

Boundary conditions: what is $\{\varepsilon\}^*$ ?
what is $\emptyset^*$ ?
The Study of Languages is Important

• A fundamental computing problem:

  Given (some description of) \( L \), and \( w \), is \( w \) in \( L \)?

• Examples

  – \( H = \{<G> \mid <G> \text{ encodes a graph that contains a Hamiltonian cycle}\} \)
  
  – \( G = \{n \mid n \text{ is even and is the sum of two primes}\} \)
    
    Goldbach’s conjecture: \( G = \{\text{all even numbers} > 2\} \)
  
  – \( P = \{\rho \mid \rho \text{ is a Python program that for any input will terminate properly}\} \)
Not all languages are alike

- Different languages have different properties
- Worth exploring all the languages with a given property

Examples:
- Finite languages (boring)
- Computable languages (interesting)
- Poly-time-decidable languages (interesting)
- NP-complete languages
Regular Languages

The set of *regular languages* over some alphabet $\Sigma$ is defined inductively by:

- $\emptyset$ is a regular language
- $\{\varepsilon\}$ is a regular language
- $\{a\}$ is a regular language for each $a$ in $\Sigma$
- If $L_1$, $L_2$ are regular, then $L_1 \cup L_2$ is regular
- If $L_1$, $L_2$ are regular, then $L_1L_2$ is regular
- If $L$ is regular, then $L^*$ is regular
Regular Languages Examples

- If $L = \{w\}$ for a single string $w$, then $L$ is regular
  - Example: $L = \{w\} = \{aba\}$.
    Then $\{aba\} = \{a\}\{b\}\{a\}$, and $\{a\}$, $\{b\}$ are regular

- If $L$ is any finite set of strings then $L$ is regular
  - Example: $L = \{w_i\mid w_i$ is the $i$-th prime, $i < 100\}$
    Then for each $i$, $\{w_i\}$ is regular, and $L$ can be expressed as the repeated (but finite) union of regular languages $\{w_i\}$
Regular Languages Examples

• \{w \mid w \text{ is a keyword or identifier in a Python program}\}
• \{w \mid w \text{ describes a valid Roman numeral}\}
  e.g., \{I, II, III, IV, V, VI, VII, VIII, IX, X, ..., XLVII, etc.\}
• \{w \mid w \text{ is a valid date of the form mm/dd/yy}\}
• \{w \mid w \text{ is a valid sequence of button-pushes to ready the A/V system for CS 374 lectures}\}
Regular Expressions
Regular Expressions

- a way to denote the regular languages
- simple *patterns* to describe related strings
- useful in
  - text search (editors, Unix/grep)
  - compilers: lexical analysis
  - compact way of representing sets of strings
- dates back to 50’s: Stephen Kleene, who has a star named after him*

* The star named after him is the Kleene star “*”
Inductive Definition

A regular expression $r$ over alphabet $\Sigma$ is one of the following:

Base cases

- $\emptyset$ denotes the language $L(\emptyset) = \emptyset = \{\}$
- $\epsilon$ denotes the language $L(\epsilon) = \{\epsilon\}$
- $a$ for $a$ in $\Sigma$, denotes the language $L(a) = \{a\}$
Inductive Definition

A regular expression \( r \) over alphabet \( \Sigma \) is one of the following:

**Inductively defined cases**

If \( r_1 \) and \( r_2 \) are regular expressions denoting languages \( R_1 \) and \( R_2 \), then

- \((r_1 + r_2)\) is a regular expression denoting \( R_1 \cup R_2 \)
- \((r_1r_2)\) is a regular expression denoting \( R_1R_2 \)
- \((r_1)^*\) is a regular expression denoting \((R_1)^*\)
Compare with regular languages

**REGULAR LANGUAGES**

- $\emptyset$ regular
- $\{\epsilon\}$ regular
- $\{a\}$ regular for $a$ in $\Sigma$
- $R_1 \cup R_2$ regular if both are
- $R_1 R_2$ is regular if both are
- $R^*$ is regular if $R$ is.

**REGULAR EXPRESSIONS**

- $\emptyset$ denotes $\emptyset$
- $\epsilon$ denotes $\{\epsilon\}$
- $a$ denotes $\{a\}$
- $r_1 + r_2$ denotes $R_1 \cup R_2$
- $r_1 r_2$ denotes $R_1 R_2$
- $r^*$ denotes $R^*$

*Regular expressions denote regular languages*
*(they show the operations used to form the language)*
Parentheses

- Omit parentheses by adopting precedence order: \(*\), concat, \(+\). E.g., \(r^*s + t = ((r^*)s)+t\)
- Omit parentheses by associativity of each of these operations. E.g., \(rst = (rs)t = r(st)\)

Superscript +

- For convenience, define \(r^+ = rr^*\)
  so if \(r\) denotes language \(R\), then \(r^+\) denotes \(R^+\)

Other notation

- \(r + s\), \(r U s\), and \(r|s\) all denote the “or” or union
- \(rs\) is sometimes written \(r \bullet s\)
Examples

• \((0+1)^*001(0+1)^*\)
  – strings with 001 as a substring

• \(0^*+ (0^*10^*10^*10^*)^*\)
  – strings with a number of 1’s divisible by 3

• \(\emptyset 0\)
  – concatenation of anything in here \{ \} with anything in here \{0\}, so = \{ \} = \emptyset (no strings may be so formed)

• \((\varepsilon +1)(01)^*(\varepsilon +0)\)
  – alternating 0s and 1s

• \((\varepsilon +0)(1+10)^*\)
  – strings without two consecutive 0s
Challenge: create regular expressions

- bitstrings with either the pattern 001 or the pattern 100 occurring somewhere
  one answer: \((0+1)^*001(0+1)^* + (0+1)^*100(0+1)^*\)

- bitstrings with an odd number of 1s
  one answer: \(0^*10^*(0^*10^*10^*)^*\)

Real challenge: bitstrings with an odd number of 1s AND an odd number of 0s
**Regular Expression Identities**

- $r^*r^* = r^*$
- $(r^*)^* = r^*$
- $rr^* = r^*r$
- $(rs)^*r = r(sr)^*$
- $(r+s)^* = (r^*s^*)^* = (r^* + s^*)^* = (r+s)^* = ...$
Define $L$ over $\{0,1\}^*$ by:
- $\varepsilon$ is in $L$
- if $w$ is in $L$, then $0w1$ is in $L$

What do strings in $L$ look like?
Give a characterization of $L$ and prove it correct.
Can you find a regular expression for $L$?
Conjecture: \[ L = \{0^i1^i : i \geq 0\} \]

How can we prove this is correct?

Prove (by induction) that

(a) \[ L \subseteq \{0^i1^i : i \geq 0\} \]
(b) \[ L \supseteq \{0^i1^i : i \geq 0\} \]
\[ L \subseteq \{0^i1^i : i \geq 0\} \]

Show by induction on \(|w|\), that if \(w\) is in \(L\), then \(w\) is of the form \(0^i1^i\).

Base case: \(|w| = 0\).

Then \(w = \varepsilon = 0^01^0\)

Let \(n > 0\), and assume for all \(k < n\) that

for any \(w\) in \(L\) with \(|w| = k\), \(w\) is of form \(0^i1^i\)
Inductive step

Now consider arbitrary $w$ in $L$, with $|w| = n$.

Then $w = 0u1$ where $u$ in $L$ has size $n - 2 < n$
(by definition of $L$)

By induction, $u$ is of form $0^i1^i$.

Then $w = 0u1 = 00^i1^i1 = 0^{i+1}1^{i+1}$, the required form
$L \supseteq \{0^i1^i : i \geq 0\}$

Show by induction on $|w|$, that if $w$ is of the form $0^i1^i$, then $w$ is in $L$.

**Base case:** $|w| = 0$.

Then $w = 0^01^0 = \varepsilon$, which is in $L$ by definition.

**Inductive step:**
Let $n > 0$, and assume for all $k < n$ that $0^k1^k$ in $L$

$0^n1^n = 00^{n-1}1^{n-1}1 = 0u1$, with $u$ in $L$ by induction

Since $u$ in $L$, so is $0u1 = 0^n1^n$ by definition of $L$