Languages, Regular Expressions, and Finite Automata

Alphabets and Strings

- an *alphabet* is a finite, non-empty set of *symbols*
- a *string* over an alphabet is a finite sequence of symbols from that alphabet
  - a sequence – the order matters
  - two strings are equal only if they have exactly the same symbols in the same order (implies that they have the same length)
- convention
  - letters from the beginning of the English alphabet (*a*, *b*, *c*, etc) refer to individual symbols
  - letters from the end of the alphabet (*u*, *v*, *w*, etc) refer to strings

String Operations

- *length* is the number of symbols, written $|x|$
- *concatenation* appends one string to another, written *xy*
  - associative – $(xy)z = x(yz)$
  - not commutative – $xy \neq yx$ unless $x = y$ or $x = \varepsilon$ and/or $y = \varepsilon$
- the *reverse* string contains the same symbols in the opposite order, written x^R
- the *empty string* ε (sometimes written λ) contains no symbols
  - $|\varepsilon| = 0$
  - $\varepsilon^R = \varepsilon$
  - $\varepsilon x = x\varepsilon = x$

Languages

- Σ^* is the set of strings made up of 0 or more symbols from alphabet Σ i.e. the set of all strings over Σ
  - Σ^* is countably infinite
    - list the strings in the order of strings with 0 symbols, strings with 1 symbol, strings with 2 symbols, etc – each group of length *k* strings is finite
- a *language* over alphabet Σ is a subset of Σ^*
  - a language over Σ is an element of $\mathcal{P}(\Sigma^*)$ – any set of strings over Σ is a language over Σ
- a language can be finite or infinite
- there are an uncountable number of languages over Σ

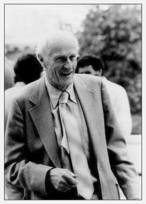
Operations on Languages

- languages are sets, so ∪, ∩, and ‾ (complement) operations apply
- the *concatenation* of two languages *S*, *T*

  $ST = \{ st \mid s \in S \wedge t \in T \}$

  – like the concatenation of strings, associative but not commutative
- S^k = language *S* concatenated to itself *k* times i.e. the set of strings formed from *k* strings of *S*
  – $S^0 = \{\varepsilon\}$ – the set of strings formed from 0 strings
- the *Kleene closure* $S^* = S^0 \cup S^1 \cup S^2 \cup \ldots$ is the set of all strings formed from concatenating 0 or more strings from *S*
  – * = *Kleene star*

Stephen Kleene



- 1909-1994
- American mathematician
- last name commonly pronounced KLEE-nee or KLEEN
- Kleene pronounced it KLAY-nee
- known for
  – recursion theory (a branch of mathematical logic) – Kleene's recursion theorem
  – contributions to the foundations of theoretical computer science
  – Kleene hierarchy, Kleene algebra, Kleene fixed-point theorem
  – regular expressions

1. Let $S = \{\varepsilon, ab, abab\}$ and $T = \{aa, aba, abba, abbba, \ldots\}$. Find the following:
a) S^2 **b)** S^3 **c)** S^* **d)** ST **e)** TS

2. The ***reverse*** of a language L is defined to be $L^R = \{x^R \mid x \in L\}$. Find S^R and T^R for the S and T in the preceding problem.

3. Give an example of a language L such that $L = L^*$.

– if $L = L^*$, L must
  - contain ε
  - be closed under concatentation

Regular Expression

- a *regular expression* is a specific kind of pattern that describes strings with a certain form

Regular Expressions

Definition 3.2. Let Σ be an alphabet. Then the following patterns are ***regular expressions*** over Σ:

fee or fi? the Greek pronunciation of Φ is fee, but fi is common in (US) English (and math)

1. Φ and ε are regular expressions;
2. a is a regular expression, for each $a \in \Sigma$;
3. if r_1 and r_2 are regular expressions, then so are $r_1 \mid r_2$, $r_1 \cdot r_2$, r_1^* and (r_1) (and of course, r_2^* and (r_2)). As in concatenation of strings, the $\cdot$ is often left out of the second expression. (Note: the order of precedence of operators, from lowest to highest, is $\mid$, $\cdot$, $*$.)

No other patterns are regular expressions.

- so far this only describes the syntax of a regular expression – what sequences of symbols one can write down to form a regular expression (the meaning of these symbols is next)

Regular Expressions

Definition 3.3. The ***language generated by a regular expression*** r, denoted $L(r)$, is defined as follows:

1. $L(\Phi) = \emptyset$, i.e. no strings match Φ;
2. $L(\varepsilon) = \{\varepsilon\}$, i.e. ε matches only the empty string;
3. $L(a) = \{a\}$, i.e. a matches only the string a;
4. $L(r_1 \mid r_2) = L(r_1) \cup L(r_2)$, i.e. $r_1 \mid r_2$ matches strings that match r_1 or r_2 or both;
5. $L(r_1 r_2) = L(r_1)L(r_2)$, i.e. $r_1 r_2$ matches strings of the form "something that matches r_1 followed by something that matches r_2";
6. $L(r_1^*) = (L(r_1))^*$, i.e. r_1^* matches sequences of 0 or more strings each of which matches r_1.
7. $L((r_1)) = L(r_1)$, i.e. (r_1) matches exactly those strings matched by r_1.

- this defines what a given regular expression means

1. Give English-language descriptions of the languages generated by the following regular expressions.
a) $(a \mid b)^*$ **b)** $a^* \mid b^*$ **c)** $b^*(ab^*ab^*)^*$ **d)** $b^*(abb^*)$

2. Give regular expressions over $\Sigma = \{a, b\}$ that generate the following languages.
a) $L_1 = \{x \mid x \text{ contains 3 consecutive } a\text{'s}\}$
b) $L_2 = \{x \mid x \text{ has even length}\}$
c) $L_3 = \{x \mid n_b(x) = 2 \bmod 3\}$
d) $L_4 = \{x \mid x \text{ contains the substring } aaba\}$
e) $L_5 = \{x \mid n_b(x) < 2\}$
f) $L_6 = \{x \mid x \text{ doesn't end in } aa\}$