Chapter 2

Languages

Languages

- Defn. A language is a set of strings over an alphabet.
  - A more restricted definition requires some forms of restrictions on the strings, i.e., strings that satisfy certain properties
- Defn. The syntax of a language restricts the set of strings that satisfy certain *properties*.

Languages

- Defn. A string over an alphabet *X*, denoted Σ, is a *finite sequence* of elements from *X*, which are *indivisible objects*
  - e.g., Strings can be words in English

➢ The set of strings over an alphabet is defined recursively (as given below)

Languages

- Defn. 2.1.1. Let Σ be an alphabet. Σ^*, the set of strings over Σ, is defined *recursively* as follows:

  (i) *Basis*: $\lambda \in \Sigma^*$, the *null string*

  (ii) *Recursion*: $w \in \Sigma^*$, $a \in \Sigma \Rightarrow wa \in \Sigma^*$

  (iii) *Closure*: $w \in \Sigma^*$ is obtained by step (i) and a finite # of step (ii)

  - The length of a string *w* is denoted *length(w)*

- Q: If Σ contains *n* elements, how many possible strings over Σ are of length *k* ($\in \Sigma^*$)?

Languages

- Example: Given $\Sigma = \{a, b\}$, Σ^* includes λ, *a*, *b, aa, ab, ba, bb, aaa*, …
- Defn 2.1.2. A language over an alphabet Σ is a subset of Σ^*.
- Defn 2.1.3. *Concatenation*, is the fundamental *binary* operation in the generation of strings, which is *associative*, but not *commutative*, is defined as
  i. Basis: If $length(v) = 0$, then $v = \lambda$ and $uv = u$
  ii. Recursion: Let *v* be a string with $length(v) = n$ (> 0). Then $v = wa$, for string *w* with length n-1 and $a \in \Sigma$, and $uv = (uw)a$

Languages

- Example: Let $\alpha = ab$, $\beta = cd$, and $\gamma = e$
  - $\alpha(\beta\gamma) = (\alpha\beta)\gamma$, but
  - $\alpha\beta \neq \beta\alpha$, unless $\alpha = \lambda$, $\beta = \lambda$, or $\alpha = \beta$.
- Exponents are used to abbreviate the *concatenation* of a string with itself, denoted u^n ($n \geq 0$)
- Defn 2.1.5. Reversal, which is a unary operation, rewrites a string *backward*, is defined as

  i) Basis: If length(u) = 0, then $u = \lambda$ and $\lambda^R = \lambda$.

  ii) Recursion: If length(u) = n (> 0), then $u = wa$ for some string w with length n - 1 and some $a \in \Sigma$, and $u^R = aw^R$
- Theorem 2.1.6. let $u, v \in \Sigma^*$. Then, $(uv)^R = v^R u^R$.

Languages

- Finite language specification
  - Example 2.2.1. The language L of string over $\{a, b\}$ in which each string begins with an 'a' and has even length.
    - i) Basis: $aa, ab \in L$.
    - ii) Recursion: If $u \in L$, then $uaa, uab, uba, ubb \in L$.
    - iii) Closure: $u \in L$ only if u is obtained from the basis elements by a finite number of applications of the recursive step.
- Use *set operations* to construct complex sets of strings.
  - Defn 2.2.1. The *concatenation* of languages *X and Y*, denoted XY, is the language

$$XY = \{ uv \mid u \in X \text{ and } v \in Y \}$$

  - Given a set X, X^* denotes the set of strings that can be defined with $\bullet$ and $\cup$

Languages

- Defn 2.2.2. let X be a set. Then

$$X^* = \bigcup_{i=0}^{\infty} X^i \quad \text{and} \quad X^+ = \bigcup_{i=1}^{\infty} X^i$$

  - $X^+ = XX^*$ or $X^+ = X^* - \{ \lambda \}$

- Observation: Formal (i) *recursive* definitions, (ii) *concatenation*, and (iii) *set operations* precisely define *languages*, which require the unambiguous specification of the strings that belong to the language.

Regular Sets and Expressions

- Defn 2.3.1 Let Σ be an alphabet. The regular sets over Σ are defined recursively as follows:

  (i) *Basis*: $\emptyset$, $\{ \lambda \}$, and $\{ a \}$, $\forall_{a \in \Sigma}$, are *regular sets* over Σ.

  (ii) *Recursion*: Let X and Y be *regular sets* over Σ. The sets $X \cup Y$, XY and X^* are *regular sets* over Σ.

  (iii) *Closure*: Any *regular set* over Σ is obtained from (i) and by a finite number of applications of (ii).

- Example: Describe the content of each of the following regular sets:

  (i) $\{ aa \}^*$, (ii) $\{ a \}^* \cup \{ b \}^*$, (iii) $(\{a\} \cup \{b\})^*$, (iv) $\{ a \} (\{b\}\{c\})^*$ ▶

  - Regular expressions are used to *abbreviate* the descriptions of regular sets, e.g., replacing $\{ b \}$ by b, union ($\cup$) by (,), etc. ▶

Languages

<u>Examples</u>.

- (a) The set of strings over { *a*, *b* } that contains the substrings *aa* or *bb*

  $L = \{\{a\} \cup \{b\}\}^*\{a\}\{a\}\{\{a\} \cup \{b\}\}^* \cup \{\{a\} \cup \{b\}\}^*\{b\}\{b\}\{\{a\} \cup \{b\}\}^*$

- (b) The set of string over { *a*, *b* } that do <u>not</u> contain the substrings *aa* and *bb*

  $L = (a, b)^* - ((a, b)^*aa(a, b)^* \cup (a, b)^*bb(a, b)^*)$ [non-regular set]

- (c) The set of strings over {*a*, *b*} that contain exactly two *b*'s

  $L = \{a\}^*\{b\}\{a\}^*\{b\}\{a\}^*$

Regular Sets and Expressions

- Defn 2.3.2. let Σ be an alphabet. The regular expressions over Σ are defined recursively as follows:

  (i) *Basis*: $\varnothing$, λ, and a, $\forall_{a \in \Sigma}$, are *regular expressions* over Σ.

  (ii) *Recursion*: Let u and v be *regular expressions* over Σ. Then (u, v), (uv) and $(u)^*$ are *regular expressions* over Σ.

  (iii) *Closure*: Any regular expression over Σ is obtained form (i) and by a finite number of applications of (ii).

  - It is assumed that the following precedence is assigned to the operators to reduce the number of parentheses:

$$*,\ \bullet,\ \cup$$

Regular Sets and Expressions

- Example: Give a regular expression for each of the following over the alphabet { 0, 1 }:
  - { *w* | *w* begins with a '1' and ends with a '0' }
  - { *w* | *w* contains at least three 1's}
  - { *w* | *w* is any string without the substring '11' }
  - { *w* | *w* is a string that begin with a '1' and contain exactly two 0's }
  - { *w* | *w* contains an even number of 0's, or contains exactly two 1's and nothing else }
- Regular expression definition of a language is not unique.

Regular Expression Identities

TABLE 2.1	Regular Expression Identities
1.	$\phi u = u\phi = \phi$
2.	$\lambda u = u\lambda = u$
3.	$\phi^* = \lambda$
4.	$\lambda^* = \lambda$
5.	$u \cup v = v \cup u$
6	$u \cup \phi = u$
7.	$u \cup u = u$
8.	$u^* = (u^*)^*$
9.	$u(v \cup w) = uv \cup uw$
10.	$(u \cup v)w = uw \cup vw$
11.	$(uv)^*u = u(vu)^*$
12.	$(u \cup v)^* = (u^* \cup v)^*$ $= u^*(u \cup v)^* = (u \cup vu^*)^*$ $= (u^*v^*)^* = u^*(vu^*)^*$ $= (u^*v)^*u^*$

Regular Expressions

- There exist non-regular expressions such as
  - $\{a^n b^n \mid n \geq 0\}$
  - $\{(0, 1)^*(01)^n(0, 1)^*(10)^n(0,1)^*1 \mid n \geq 0\}$

- Table 2.1 Regular Expression Identities

  - $\phi^* = \lambda$; The * operation puts together any number of strings from the language to get a (new) string in the result. If the language is empty, the * operation can put together 0 strings, giving only the null string (λ).

  - $\phi u = u \phi = \phi$; Concatenating ϕ to any set yields ϕ.

- $(a, \lambda)(b, \lambda) = \{\lambda, a, b, ab\}$. How about $c^*(b, ac^*)^*$?

  - The regular expression $c^*(b, ac^*)^*$ yields all strings that do not contain the substring *bc*.

Grammars Languages and Accepting Machines

Grammars	Languages	Accepting Machines
Type 0 grammars, Phrase-structure grammars, Unrestricted grammars	Recursively enumerable Unrestricted	TM NDTM
Type 1 grammars, Context-sensitive grammars, Monotonic grammars	Contest-sensitive languages	Linear-bounded Automata
Type 2 grammars, Context-free grammars	Context-free languages	PDA
Type 3 grammars, Regular grammars, Left-linear grammars, Right-linear grammars	Regular languages	FSA NDFA