Theory of Computer Science – SCJ 3203

Strings and Languages

Paridah Samsuri
Mohd Soperi Mohd Zahid

Outline

- Strings and its operations
- Language Specification

Strings and Languages

- Natural languages, computer languages, Mathematical languages
- A **language** is a set of **strings** over an **alphabet**.
- Syntax of the language: certain properties that must be satisfied by strings.

Strings and Alphabets

- A String over a set X is a finite sequence of elements from X.
- The set of elements are called alphabet of the language.
- Alphabet consists of a finite set of indivisible objects.
- The alphabet of a language is denoted $\sum$

Strings and Alphabets

- String over an alphabet is a finite sequence of symbols from that alphabet, written next to one another and not separated by commas.

Example:

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

then strings over Σ are:

0 1 00 01 10 11

1010

1110100111

111111 and etc.

String Recursive Definition

Let Σ be an alphabet.

The set of strings over Σ (written as Σ^*)

is defined recursively as follows:

- **Basis** : $\lambda \in \Sigma^*$
- **Recursive step**: if $w \in \Sigma^*$ and $a \in \Sigma$

  Then

  $wa \in \Sigma^*$
- **Closure**: $w \in \Sigma^*$ if and only if it can be obtained from basis element λ by a finite number of applications of recursive step.

Length of String

- If w is a string over Σ, the length of w, written **|w|** is the number of symbols that it contains.

  Example:

  $|\lambda| = 0$

  $|0| = 1$

  $|1| = 1$

  $|1010| = 4$

  $|001101| = 6$

- The set of strings, Σ^*, includes:

length 0: λ

length 1: a b c

length 2: aa ab ac ba bb bc ca cb cc

length 3: aaa aab aac aba abb abc
aca acb acc baa bab bac
bba bbb bbc bca bcb bcc
caa cab cac cba cbb cbc
cca ccb ccc

Concatenation of String

- If we have string x of length m and string y of length n, the **concatenation** of x and y written xy is $x_1 \ldots x_m y_1 \ldots y_n$.

- Example: x = aba

  y = bbbab

  Then xy = ababbbab

  yx = bbbababa

  xyx = ababbbababa

Self Concatenation

- If we have string x, the **concatenation** of x and x is **self concatenation**
- Example:

$x^0 = \lambda$

$x^1 = x = aba$

$x^2 = xx = abaaba$

$x^3 = xxx = abaabaaba$

- $x^k = xx\ldots x$: self-concatenated string k times

Definition of Concatenation

Let u, v $\in \Sigma^*$.

The concatenation of u and v, written as uv,

is a binary operation on Σ^* defined as follows:

- **Basis** : length (v)=0, then v=λ and uv=u
- **Recursive step**:

  Let v be a string with length(v)=n>0.

  Then v=wa, and uv=(uw)a

  for some string w with length n-1 and a$\in\Sigma$

Let u=ab, v=ca, w=bb.

Then the concatenation of:

uv=abca

vw=cabb

(uv)w=abcabb

u(vw)=abcabb

The result is independent of the order in which the operations are performed.

Substring

- String z is a **substring** of w if z appears consecutively within w

  (z occurs inside of w)

  Example:

  w = abbaaababb

  bba   is a substring of w

  abab   is a substring of w

  baba   is NOT a substring of w

Prefix

Prefix for string **v** is a substring **u** where :

if **x** **=** λ, for ν **=** **xuy**

Then ν **= uy**

(string v **starts with** substring u)

Example:

if w = abbaaababb

a is a prefix of w

abbaa is a prefix of w

bba is NOT a prefix of w

Suffix

Suffix for string **v** is a substring **u** where :

if $\mathbf{y} = \lambda$, for $\nu = \mathbf{xuy}$

Then $\nu = \mathbf{xu}$

(string v **ends with** substring u)

Example:

if w = abbaaababb

abb	is a suffix of w
babb	is a suffix of w
bab	is NOT a suffix of w

Reverse

The **reverse** of w, written w^R or w^r is the string obtained by writing w in the opposite order.

Example:

if $w = a$ $\quad w^R = a$

if $w = abb$ $\quad w^R = bba$

if $w = aba$ $\quad w^R = aba$

if $w = abbcd$ $\quad w^R = dcbba$

Definition of Reverse

Let u be a string in Σ^*

The reverse of u, written as $\mathbf{u}^{\mathbf{R}}$

is defined recursively as follows:

- **Basis** : length(u)=0

  Then u= λ and $\lambda^{\mathbf{R}} = \lambda$

- **Recursive step**: if w$\in\Sigma^*$ and a$\in\Sigma$

  if length(u)=n>0,

  then u=wa and $u^{\mathbf{R}} = aw^{\mathbf{R}}$

  for some string w with length n-1 and some a$\in\Sigma$

Finite Spec. of Languages

- Two ways of specifying languages:
  - an alphabet and the exhaustive list of all valid words
  - an alphabet and a set of rules defining the acceptable words.

- Definition of a new language PALINDROME over the alphabet :

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

$$\text{PALINDROME} = \{\varepsilon, \text{and all string x such that reverse(x)} = \text{x}\}$$

- if we begin listing the elements in PALINDROME, we find

$$\text{PALINDROME} = \{\varepsilon \quad \text{a} \quad \text{b} \quad \text{aa} \quad bb \quad aaa \quad aba \quad bab \quad bbb \quad aaaa \quad abba \ldots\}$$

which if we concatenate 2 words in PALINDROME, sometimes it can produce a new words which is also in PALINDROME but sometimes it doesn't. (Talk about it later)

- Closure of the alphabet (Σ) is a language in which any string of letters from an alphabet Σ^* is a word. For example,
  - if $\Sigma = \{x\}$, then

$$\Sigma^* = L_4 = \{\varepsilon \quad x \quad xx \quad xxx \ldots\}$$

  - if $\Sigma = \{0 \quad 1\}$,then

$$\Sigma^* = \{\varepsilon \quad 0 \quad 1 \quad 00 \quad 01 \quad 10 \quad 11 \quad 000 \quad 001 \ldots\}$$

- We can think of the Kleene star as

  an operation that makes an infinite language of strings of letter out of an alphabet.

  Infinite language = infinitely many words, each of finite length.

- If S is a set of words, then by **S***

  we mean the set of all finite strings formed by concatenating words from S,

  where any word may be used as often as we like, and where the null string is also included.

- If S = {aa b}, then

S* = {λ plus any word composed of
factors of aa and b}

= {λ plus all strings of a's and b's in
which the a's occur in even clumps}

= {λ b aa bb aab baa bbb . . . }

- If S = {a ab}, then

S* = {λ plus any word composed of factors of a and ab}

= {λ plus all strings of a's and b's except those that start with b and those that contain a double b}

= {λ a aa ab aaa aab aba . . . }

- To prove that a certain word is in the closure language S*, we must show how it can be written as a concatenate of words from the base set S.
- For example,
  - to show abaab is in S*, we can factor it as (ab)(a)(ab) and these are in S, therefore, their concatenation is in S*.
- If there is only one way to factor the string, we say that the factoring is unique.

- Consider the 2 languages

$$S = \{a \;\; b \;\; ab\} \quad \text{and} \quad T = \{a \;\; b \;\; bb\}$$

both S* and T* are languages of all strings of a's and b's since any string of a's and b's can be factored into syllables of either (a) or (b), both of which are in S and T.

- If we would like to refer to only the concatenation of some (not zero) strings from a set S, we use the notation + instead of *, for example:

if $\Sigma = \{x\}$,

then $\Sigma^{+} = \{x \quad xx \quad xxx \ldots\}$

- if S is a set of strings not include λ, then S^+ is the language S* without the word λ.
- If S is a language that does contain λ, then $S^+ = S^*$.
- S+ can contain λ only when S contains the word ε initially.

Theorem 1:

for any set S of strings we have **S* = S****

Proof:

every word in S** is made up of factors from S*. Every factor from S* is made up of factors from S. Therefore, every word in S** is made up of factors from S. Therefore, every word in S** is also a word in S*.

$$S^{**} \subset S^{*}$$

(is contained in or equal to)

- we can describe a language definition looked similar to

$$L_1 = \{x^n \quad \text{for } n = 1 \;\; 2 \;\; 3 \ldots\}$$

$$L_2 = \{x^n \quad \text{for } n = 1 \;\; 3 \;\; 5 \ldots\}$$

- we can guess the meaning of the languages, however it can be defined in a particular way that gets hard to guess. For example:

$$L_6 = \{x^n \quad \text{for } n = 3 \;\; 4 \;\; 8 \;\; 22 \ldots\}$$

language definition

- We shall develop some new language-definition symbolism that will be much more precise than the …

For example: consider the language L_4

$$L_4 = \{\varepsilon \quad x \quad xx \quad xxx \quad xxxx \; K \}$$

We can define it with closure

Let $S = \{x\}$ Then $L_4 = S^*$

for shorthand, we could have written $L_4 = \{x\}^*$

- By using Kleene star, we can have a simple expression instead of …
- In order to distinguish between x from alphabet of x from Kleene star x*, we will use bold face **x*** instead to make it different.

$$\mathbf{x^*} = \varepsilon \text{ or } \mathrm{x} \text{ or } \mathrm{x}^2 \text{ or } \mathrm{x}^3 \text{ or } \mathrm{x}^4 \mathrm{K}$$

$$= x^n \text{ for some } \mathrm{n} = 0 \;\; 1 \;\; 2 \;\; 3 \;\; 4\mathrm{K}$$

- We can also define L_4 as

$$L_4 = \text{language}(\,\mathbf{x}^*\,)$$

Since x* is any string of x's, L_4 is then the set of all possible string of x's of any length (including λ)

- Suppose we wish to describe the language L over the alphabet $\Sigma = \{a \quad b\}$

where $L = \{a \quad ab \quad abb \quad abbb \quad abbbb \ldots\}$

"all words of the form one a follow by some number of b's (maybe no b's at all)"

we may write $L = \text{language}(\mathbf{ab^*})$

$$(\mathbf{ab})^* = \quad \varepsilon \text{ or } \text{ab} \text{ or } \text{abab} \text{ or } \text{ababab} \ldots$$

- Parentheses are not letters in the alphabet of this language, so they can be used to indicate factoring without accidentally changing the words.
- Like the powers in algebra

ab* means a(b*), not (ab)*

$$L_1 = \text{language}(\mathbf{xx}^*)$$

means ?

We start each word of L_1 by writing down an x and then we follow it with some string of x's (which may be no more x's at all.)

We can use the $^+$ notation and write

$$L_1 = \text{language}(\mathbf{x}^+)$$

- The language L1 defined above can also be defined by any of these expressions:

$$xx^*x^+ \qquad xx^*x^* \qquad x^*xx^*$$

$$x^+x^* \qquad x^*x^+ \qquad x^*x^*x^*xx^*$$

Remember

x^* can always be λ

ab*a

is the set of all string of a's and b's that have at least two letters, that begin and end with a's, and that
have nothing but b's inside
(if anything at all).

$$\text{Language}(\mathbf{ab}^*\,\mathbf{a}) = \{\text{aa} \quad \text{aba} \quad \text{abba} \quad \text{abbba} \quad \text{abbbba} \ldots\}$$

a*b*

contains all the strings of a's and b's in which all the a's (if any) come before all b's (if any)

$$Language(\mathbf{a}^*\mathbf{b}^*) = \{\varepsilon \quad a \quad b \quad aa \quad ab \quad bb \quad aaa \quad aab \quad abb \quad bbb \quad aaaa \ldots\}$$

notice that

ba and **aba** are not in this Language

$$a^* b^* \neq (ab)^*$$

(ab)* can contain abab

but

a*b* can't contain abab

- Let A and B be languages. We define the regular operations union, concatenation, and star as follows.
  - Union :

$$A \cup B = \{x \mid x \in A \text{ or } x \in B\}$$

  - Concatenation : (simply no written)

$$A \circ B = \{xy \mid x \in A \text{ and } y \in B\}$$

  - Star :

$$A^* = \{x_1 x_2 x_3 \ldots x_k \mid k \geq 0 \text{ and each } x_i \in A\}$$

$\mathbf{x \cup y}$ where x and y are strings of characters from an alphabet

means

“either x or y”

Also written as **x+y**

- Consider the language T defined over the alphabet $\Sigma = \{a \quad b \quad c\}$

$$T = \{a \quad c \quad ab \quad cb \quad abb \quad cbb \quad abbb \quad cbbb \quad abbbb \quad cbbbb \ldots\}$$

all the words in T begin with an a or a c and then are followed by some number of b's.

$$T = language((\mathbf{a} \cup \mathbf{c})\mathbf{b}^*)$$

$$= language(\textit{either}\ \text{a or c then some b's})$$

- We can define any finite language by our new expression.
- For example,

  consider a finite language L contains all the strings of a's and b's of length 3 exactly:

$$L = \{aaa \quad aab \quad aba \quad abb \quad baa \quad bab \quad bba \quad bbb\}$$

- The first letter can be either a or b. so do the 2nd and 3rd letter.

$$L = \text{language}((\mathbf{a} \cup \mathbf{b})\ (\mathbf{a} \cup \mathbf{b})\ (\mathbf{a} \cup \mathbf{b}))$$

or we can simply write shortly as

$$L = \text{language}(\mathbf{a} \cup \mathbf{b})^3$$

if we write **(a∪b)***, it means the set of all possible strings of letters from the alphabet $\Sigma = \{a \; b\}$ including the null string λ

- If we write

$$a(a\cup b)^*$$

we can describe all words that begin with the letter a.

- If we would like to describe all words that begin with an a and end with b, we can define by the expression

$$a(a\cup b)^*b = a(\text{arbitrary string})b$$

References