Week 2

Symbols, Alphabets, Strings and Languages

Symbol – also known as characters, any single indivisible unit

Alphabet – finite set of symbols, represented by Σ

eg. Σ = {LISTEN, LOOK, CLOSE} or Σ = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, .}

String – finite sequence of symbols from an alphabet, represented as *s* or *w*, empty string = ε

eg. *s* = 4.124.567 or *s* = 126322, or *w* = LISTENCLOSE or *s* = CLOSELISTENLOOK

all possible strings:

where Σ {0,1,2,3,4,5,6,7,8,9,.}

Σ\* = {0,1,2,3,4,5,6,7,8,9,.,00,01,02,…,324.432,…}

Language – subset of all possible strings Σ\* that can be built from an alphabet, can be Ø (empty), finite or infinite

eg. where Σ = {a, b, c, …, z, A, B, C, …, Z}   
*L­­1 =* {a}, *L2* = {ab, bc, cd, …} *L3* = {all English words} *L4* = {all one-syllable words} ⊂ *L3*

Operations on Strings

Length of a string = |*w*|, |ε| = 0,

Σ = {*a, b, …, Y, Z*} |*abcd*| = 4

Σ = {*LISTEN, LOOK, CLOSE*} |*CLOSELISTEN*| = 2

Reverse string = *wR*

eg. Σ = {*LISTEN, LOOK*} *w* = *LOOKLISTEN wR* = *LISTENLOOK ~~w~~~~R~~* ~~=~~ *~~KOOLNETSIL~~*

Concatenation = *w1w2*

eg. Σ = {*1, 2, 3, 4, 5, 6, 7, 8, 9, 0, .*} *w1* = 123,*w2* = .92

*w1w2* = 123.92 *w2w1 =* .92123

Replication = concatenation repeated *n* times, represented with *wn*

eg. *w0* = ε  
 *w1* = *w*  
 *w2* = *ww*

Operations on Languages

Remember: languages are sets, so use set operations (Union, Intersection, Negation, Set difference)

Negation = full space is Σ\*

Reverse = reverse all symbols, creating new language

*L* = {*ab, bc, cd, de, ef*}  
 *LR* = {*ba, cb, dc, ed, fe*}

Concatenation = merge each pair (or more) of member strings from the first language to second (or left to right)

*L1* = {*a, b, c*}  
 *L2* = {*d, e, f*}  
 *L1L2* = {*ad, ae, af, bd, be, bf, cd, ce, cf*}  
 *L2L1* = {*da, db, dc, ea, eb, ec, fa, fb, fc*}

Kleene star = concatenation of one or more strings in a language, including the empty string, infinitely, represented with *L*\*

Kleene plus = *L+* = *LL\**

if ε ∈ *L* if ε ∉ *L  
 L+* = *L\** { ε } u *L+*

Decision Problems And Procedures

Decision problem – problem where answer if yes or no

Decision procedure – method for solving a decision problem (i.e. with a boolean)

Any problem is either a decision problem or can be recast as such

Determinism and Non-Determinism

Determinism – behaviour of an algorithm or procedure is exactly the same on a certain input

Non-determinism – behaviour of an algorithm or procedure is different with a certain input, for example if an algorithm chooses a single randomly selected output from a set of possibilities.

Prefix/suffix examples:

‘aa’ is a prefix of ‘aaba’ but not ‘baba’  
‘aa’ is a suffix of ‘abaa’ but not ‘baba’   
‘aa’ is a proper prefix and suffix of ‘aabaa’ but not ‘aa’  
every string is a prefix/suffix of itself  
ε is a prefix/suffix of every string  
substring(x,y) = true if ∃*w1, w2* ∈ Σ\* : *y* = *w1xw2*prefix(x,y) = true if ∃*w*∈ Σ\* : y = *xw*suffix(x,y) = true if ∃*w*∈ Σ\* : y = *wx*

Kleene Operators

*L\* = L+* ⋃ {ε}

String Encoding

<*X*>is the string encoding of object *X.*<*X, Y*> is the string encoding of object pair *X, Y.*

1. Σ\* = {ε, a, b, c, ab, ac, bc, abc, aab, aac, aba, …} = Countably infinitely many
2. No, the language must contain strings with equal amounts of 1’s and 2’s.
   1. No, there can only be an empty string if n = 0 for any value.
   2. Yes, this holds true when n = 2 in both languages.
   3. No, there must be an equal number of a’s and b’s in any string.
   4. Yes, this holds true when n = 2 in *L1* and n = 4 in *L2*.
3. *L\* = (w* : *w* = ε v ∃*n > 0 (*∃*w1, w2, w3, … wn* ∈ *L)*
4. Lexicographic = shortest item first, or alphabetic where lengths are equal.  
   { apple, peach, cherry. applepie, peachpie, cherrypie, applecobbler, peachcobbler, cherrycobbler }
5. *L1* x *L2 =* {(peach, pie), (peach, cobbler), (peach, ε), (apple, pie), (apple, cobbler), (apple, ε), (cherry, pie), (cherry, cobbler), (cherry, ε)}