Biçimsel Diller ve Otomata Teorisi

Sunu I

İZZET FATİH ŞENTÜRK

Languages

- In English: Letters, words, sentences
- Group of letters –> Words
- Group of words –> Sentences
- Group of sentences –> Paragraphs –> Stories ..

- Humans (mostly) agree on which sequences are valid and which are not. How?

Computer Language

- Certain character strings are recognizable words (DO, IF, END, ..)
- Certain strings -> commands
- Certain sets of commands -> program (with/o data & that can be compiled)

Language Structure

- To construct a general theory that unifies all these examples
  - Adopt a definition of a language structure
  - Decision whether a given string of units constitues a valid larger unit
  - Not matter of guesswork
  - Based on explicitly stated rules

Purpose

- Set rules for recognizing whether an input is a valid communication
- We are not interested in what the communication means

- It is important that the program compiles
- We are not interested whether it does what the programmer intended
- <u>If it compiles, it is a valid example in the language</u>

Formal Rules

- Very hard to state all the rules for the spoken language
  - Slang, idiom, dialect, poetic metaphor, etc.

- To define a general theory of formal languages
  - Insist on <u>precise</u> rules
  - Computers are not forgiving imperfect commands

Formal Rules

- Formal: All the rules for the language are explicitly stated (what strings of symbols can occur)
  - No liberties are tolerated
  - No reference to any deep understanding is required

- Language
  - Symbols on paper not expression of ideas
  - Not communication among intellects but a gam of symbols with formal rules

Alphabet

- We begin with a finite set of fundamental units to build structures
- A certain set of strings of characters from the alphabet -> language
- Strings permissible in the language -> words
- Symbols in the alphabet do not have to be Latin letters
- Only universal requirement for a possible string: it contains only finitely many symbols

Empty/Null String

- We wish to allow a string to have no letters: empty/null string
- Denote with: Λ (Greek capital lambda)

- The null string is always Λ (no matter which alphabet used)
- The null word is always Λ (if it is a word in the language)

Comparing Words

- Two words are considered same if..
  - all their letters are the same
  - All their letters are in the same order

- There is only one possible word of no letters: Λ
- For clarity, we usually do not allow symbol Λ to be part of the alphabet for any language

The Language with no Words

- Important difference between..
  - The word that has no letters
  - Λ
  - The language that has no words (φ – (small Greek letter phi))

- It is not true that Λ is part of φ, φ has no words
- If a language L does not contain Λ, we can add it: $L + \{\Lambda\}$
  - $L + \{\Lambda\} \neq L$
  - $L + \varphi = L$

The Language with no Words

- The fact that φ is a language without any words is an important distinction
- When we have a “method” to produce a language
  - The method can <u>fail</u> and produces nothing or..
  - The method successfully produces the language φ

Defining Alphabet

- $\Sigma = \{a\ b\ c\ d\ e\ \ldots\ z\}$
- We can use spaces ot commas to separate the elements

Specifying Valid Words

- We can list all valid words – as done in a dictionary
  - Long list but finite!

- WORDS = {all the words in a standard dictionary}

- We do not allow the possibility of defining a language by an infinite dictionary

Form a Viable Sentence

- To know all the words in a finite language (English, etc) does not imply the ability to create a viable sentence
- Define a new alphabet Γ (capital gamma)
- Γ = {the entries in a standard dictionary, plus a blank space, plus the punctuation marks}
- We can never produce a complete list of all valid English sentences
  - Infinitely many words in Γ (I ate one apple, two apples, three …)
  - Finite description of an infinite language!

Grammar Rules and Meaning

- Following grammar rules of Γ only..
- I ate three Tuesdays
  - A valid word in Γ
  - We must allow this string
  - Grammatically correct
  - Meaning is ridiculous

- We are interested in <u>syntax</u> alone, no semantics!

Specifying Valid Words can be Tricky

- The language MY-PET
- The alphabet is {a c d g o t}

- There is only one word in this language
  - If the Earth and Moon ever collide then MY-PET = {cat}
  - If the Earth and Moon never collide then MY-PET = {dog}
- Not certain. Not an adequate specification of the language. Rules must enable us to decide in a finite amount of time whether a word is /not part of the language

Defining Languages

- The set of language-defining rules can be of two kinds
  - They can tell us how to test a string is a valid word or not
  - They can tell us how to construct all the words in the language

- $\Sigma = \{x\}$ An alphabet with one letter: x
- Define L_1: Any nonempty string of alphabet characters is a word
  - $L_1 = \{x\ xx\ xxx\ xxxx\ \ldots\}$ alternatively $L_1 = \{x^n \text{ for } n = 1\ 2\ 3\ \ldots\}$

Defining Languages - Concatenation

- We define the operation of concatenation
  - Two strings written side by side to form a new string
- When we concatenate xxx and xx we obtain the word xxxxx
- Analogous to addition
  - x^n concatenated with x^m: x^{n+m}
- More convenient to use new symbols other than the alphabet
  - xxx is a, xx is b and xxxxx is ab

Defining Languages - Concatenation

- It is not always true that when two words are concatenated, they produce another word in the language
- a = xxx and b = xxxxx are words in L_2 but ab is not
- In the examples ab=ba but this is not the case always!

$$\begin{aligned} L_2 &= \{x \quad xxx \quad xxxxx \quad xxxxxxx \ldots\} \\ &= \{x^{\text{odd}}\} \\ &= \{x^{2n+1} \quad \text{for} \quad n = 0 \quad 1 \quad 2 \quad 3 \ldots\} \end{aligned}$$

Defining Languages

- $\Sigma = \{0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\}$
- L_3 = {any finite string of alphabet letters that does not start with letter zero}
- L_3 looks like the set of all positive integers in base 10.
- $L_3 = \{1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10\ 11\ 12\ \ldots\}$
- If we wanted to define L_3 including word 0
- L_3 = {any finite string of alphabet letters, if it starts with a 0, has no more letters after the first}

Length Function

- We define function length of a string to be the number of letters in the string
- If a = xxxx, length(a) = 4
- If c = 428, length(c) = 3
- length(xxxxx) = 5
- length(Λ) = 0
- For any word w in any language, if length(w) = 0, w = Λ

Multiple Definitions for the Same Language

- L_3 = {any finite string of alphabet letters, if it starts with a 0, has no more letters after the first}
- One more definition of L_3
- L_3 = {any finite string of alphabet letters that, if it has length more than 1, does not start with a 0}
- Not necessarily a better definition of L_3 but illustrates that there are often different ways of specifying the same language

More on Λ

- Ambiguity in "any finite string"
  - Not clear whether Λ is part of L_3
- L_3 does not include Λ
  - We intend L_3 look like the integers
  - There is no integer with no digits
- We define L_4 = {Λ x xx xxx xxxx …}, L_4 = {x^n for n = 0 1 2 3 …}
- X^0 = Λ, not X^0 = 1 as in albegra (x^n is n x's)

Reverse Function

- We define function reverse. If a is a word in language L, then reverse(a) is the same string of letters spelled backward
  - The backward string may not be a word in L

- reverse(xxx) = xxx
- reverse(145) = 541
- reverse(140) = 041 -> 140 is a word in L_3 but not 041!

Palindrome

- We define a new language called PALINDROME over the alphabet $\Sigma = \{a\ b\}$
- PALINDROME = {Λ, and all strings x such that reverse(x) = x}
- PALINDROME = {Λ a b aa bb aaa aba bab bbb aaaa abba …}

Kleene Closure

- Given the alphabet Σ, we wish to define a language in which any string of letters from Σ is a word, even the null string. This language is called the **closure** of the alphabet

$$\Sigma^*$$

- If $\Sigma = \{x\}$ then $\Sigma^* = L_4 = \{\Lambda \ x \ xx \ xxx \ \ldots\}$
- If $\Sigma = \{0 \ 1\}$ then $\Sigma^* = \{\Lambda \ 0 \ 1 \ 00 \ 01 \ 10 \ 11 \ 000 \ 001 \ldots\}$
- If $\Sigma = \{a \ b \ c\}$ then $\Sigma^* = \{\Lambda \ a \ b \ c \ aa \ ab \ ac \ ba \ bb \ bc \ ca \ cb \ cc \ aaa \ \ldots\}$

Kleene Closure

- Kleene star is an operation that makes an infinite language of letters out of an alphabet
- Infinite language -> infinitely many words, each of finite length

Lexicographic Order

- $\Sigma^* = \{\Lambda$ a b c aa ab ac ba bb bc ca cb cc aaa …$\}$
- When we write the first several words in the language, we put them in size order (length) and then list all the words of the same length alphabetically
- This ordering is called lexicographic order
- In a dictionary, the word aardvark comes before cat. In lexicographic order it is the other way.
- If sorted alphabetically, the list would start $\{\Lambda$ a aa aaa aaaa …$\}$ would not inform us the real nature of the language

Star Operation on Words

- We can generalize the use of the star operator to sets of words, not just sets of alphabet letters
- If *S* is a set of words, then *S** is set of all finite strings formed by concatenating words from S, where any word may be formed as often as we like, where null string is also included

Star Operation on Words

- If S = {aa b} then
- S^* = {Λ plus any word composed of factors of aa and b}
- S^* = {Λ plus all strings of a's and b's in which the a's occur in even clumps}
- S^* = {Λ b aa bb aab baa bbb aaaa aabb baab bbaa bbbb aaaab aabaa aabbb baaaa baabb bbaab bbbaa bbbbb …}
- aabaaab is not is S^* since it has a clump of a's of length 3

Star Operation on Words

- Let S = {a ab} then
- S* = {Λ plus any word composed of factors of a and ab}
- S* = {Λ plus all strings of a's and b's except those that start with b and those that contain a double b}
- S* = {Λ a aa ab aaa aab aba aaaa aaab aaba abaa abab aaaaa aaaab aaaba aabaa aabab abaaa abaab ababa…}
- Double b means bb. For each word in S* every b must have an a immediately to its left. bb is impossible as it starts with a b

Star Operation on Words

- To prove that a certain word is in S^* we must show how it can written as a concatenate of words from the base set S
- In the last example, abaab is in S^*, we can factor it as follows: (ab)(a)(ab)
- These three factors are all in S, therefore, their concatenation is in S^*
- For this example, the factoring is unique. Sometimes it is not

Non-unique Factoring

- S = {xx xxx}
- *S** = {Λ and all strings of more than one x}
- *S** = {x^n for n = 0 2 3 4 5 …}
- *S** = {Λ xx xxx xxxx xxxxx xxxxxx …}
- Note that x is not in *S**
- xxxxxx is in *S** because of any of these

(xx)(xx)(xxx) or (xx)(xxx)(xx) or (xxx)(xx)(xx)

Also x^6 is either $x^2x^2x^2$ or x^3x^3

Final Remarks

- Kleene closure of two sets can end up being the same language even if the two sets that we started with were not
- S = {a b ab} and T = {a b bb}
- Both S* and T* are languages of all strings of a’s and b’s and any string of a’s and b’s can factored into syllables of either (a) or (b), both are in S and T

Final Remarks

- If we want to modify the concept of closure to refer to only the concatenation of not zero strings from a set *S* we use the notation + instead of *
- If Σ = {x}, then Σ+ = {x xx xxx …}
- If *S* is a set of strings not including Λ then *S*+ is the language *S** without the word Λ
- If *S* is a language that does not contain Λ, then *S*+ = *S**
- The plus operation is sometimes called positive closure