Regular Language

Instructor: Tom Ko

Objectives

- Chomsky hierarchy
- Regular expression and regular language

Language

- "**Language is a process of free creation**; its laws and principles are fixed, but the manner in which the principles of generation are used is **free** and infinitely varied. Even the interpretation and use of words involves a **process of free creation**." ~ Noam Chomsky, *Language and Mind*

Definitions on language in computation theory

- First, an **alphabet** is a finite set of symbols.
  - For example {*0, 1*} is an alphabet with two symbols, {*a, b*} is another alphabet with two symbols and English alphabet is also an alphabet.
- A **string** (also called a word) is a finite sequence of symbols of an alphabet.
  - *b, a* and *aabab* are examples of string over alphabet {*a, b*} and *0, 10* and *001* are examples of string over alphabet {*0, 1*}.
- A **language** is a set of strings over an alphabet.
  - Thus {*a, ab, baa*} is a language (over alphabet {*a,b*}) and {*0, 111*} is a language (over alphabet {*0,1*}).

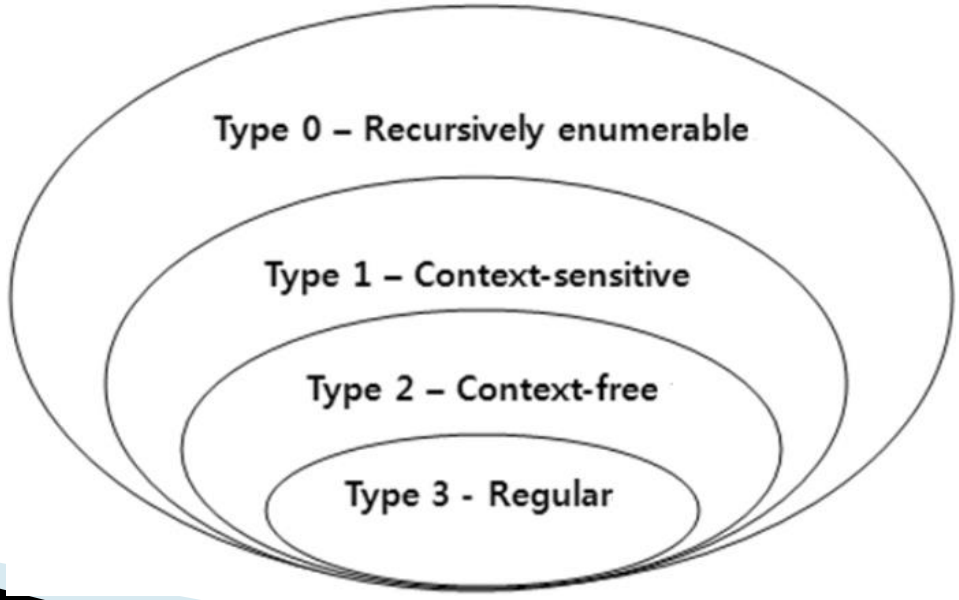
Formal language

- A **formal language** is a set of strings over a finite alphabet.
- Formal languages are not the same as natural languages.

- **Formal language theory** (FLT) is the study of formal languages, or often more accurately the study of *families* of formal languages. It deals with hierarchies of language families defined in a wide variety of ways.
- FLT is an approach of studying formal languages.
- It is closely related to automata theory, which deals with formally defined machines that *accept* (or *generate*) formal languages.
- Chomsky suggested a series of massive simplifications and abstractions to the empirical domain of natural language.

Chomsky hierarchy

- **Chomsky hierarchy** categories languages into four levels of increasing complexity.
- Regular language is the most restricted one.
- Regular and context-free languages can be described by rules.



Regular language

- Regular languages are languages which are regular enough to be simply specified by rules.
- Given an alphabet {a,b}, suppose there is a language which contains all and only the strings which start with one a's, end in two b's and must contains five letters.
- The string set will be {aaabb,aabbb,ababb,abbbb}.
- Can we have a formal way of describing such language?
  - Regular expressions
- The set of languages which can be expressed as the denotation of regular expressions is called *regular languages*

Regular expression

- Imagine now you need to replace “apple(s)” with “fruit” in your document or see all strings that look like $199 or $25 or $24.99

- Regular Expressions (RE) are used to denote/represent regular languages.
- It is a language (a sequence of characters) for specifying text search strings.
- Defining search pattern is like specifying a language.

Basic regular expression patterns

RE	Example Patterns Matched
/woodchucks/	"interesting links to woodchucks and lemurs"
/a/	"Mary Ann stopped by Mona's"
/Claire␣says,/	" "Dagmar, my gift please," Claire says,"
/DOROTHY/	"SURRENDER DOROTHY"
/!/	"You've left the burglar behind again!" said Nori

Disjunction of characters

RE	Match	Example Patterns
/[wW]oodchuck/	Woodchuck or woodchuck	"Woodchuck"
/[abc]/	'a', 'b', *or* 'c'	"In uomini, in soldati"
/[1234567890]/	any digit	"plenty of 7 to 5"

Figure 2.1 The use of the brackets [] to specify a disjunction of characters.

Range of characters

RE	Match	Example Patterns Matched
/[A-Z]/	an uppercase letter	"we should call it 'Drenched Blossoms'"
/[a-z]/	a lowercase letter	"my beans were impatient to be hoed!"
/[0-9]/	a single digit	"Chapter 1: Down the Rabbit Hole"

Figure 2.2 The use of the brackets [] plus the dash - to specify a range.

The caret ^

- There are three uses of the caret ^: to match the start of a line, as a negation inside of square brackets, and just to mean a caret

RE	Match (single characters)	Example Patterns Matched
[^A-Z]	not an uppercase letter	"Oyfn pripetchik"
[^Ss]	neither 'S' nor 's'	"I have no exquisite reason for't"
[^\.]	not a period	"our resident Djinn"
[e^]	either 'e' or '^'	"look up ^ now"
a^b	the pattern 'a^b'	"look up a^ b now"

Figure 2.3 Uses of the caret ^ for negation or just to mean ^ . We'll discuss below the need to escape the period by a backslash.

The question mark

RE	Match	Example Patterns Matched
`woodchucks?`	woodchuck or woodchucks	"woodchuck"
`colou?r`	color or colour	"colour"

Figure 2.4 The question-mark `?` marks optionality of the previous expression.

Kleene * and Kleene +

- Consider the language of sheep
  - baa!
  - baaa!
  - baaaa!
- The Kleene star means “zero or more occurrences of the immediately previous character or regular expression”
  - /a*/ means “zero or more as”
- The Kleene plus means “one or more of the previous character”
  - /a+/ means “one or more as”
- /baa+!/ for the sheep language

The period

- The period (/./) is a **wildcard** expression that matches any single character

RE	Match	Example Patterns
/beg.n/	any character between *beg* and *n*	begin, beg'n, begun

Figure 2.5 The use of the period . to specify any character.

- The wildcard is often used together with the Kleene star to mean "any string of characters".

Anchor

- **Anchor** are special characters that anchor regular expressions to particular places in a string.
- The most common anchors are the caret ^ and the dollar-sign $.
  - The caret ^ matches the start of a line.
  - The dollar sign $ matches the end of a line.
- The pattern /^The/ matches the word *The* only at the start of a line.
- \b matches a word boundary, while \B matches a non-boundary.
  - /\bthe\b/ matches the word *the* but not the word *other*

Disjunction

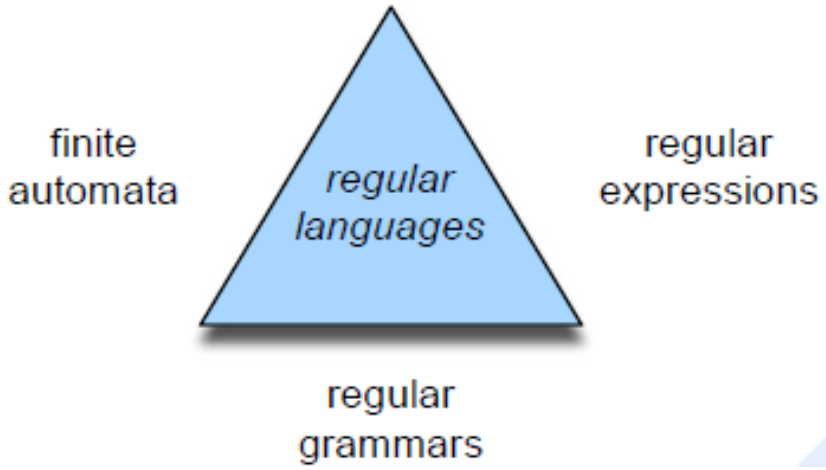
- The **disjunction** operator, also called the **pipe** symbol |.
- /cat|dog/ matches either “cat” or “dog”.
- /pupp(y|ies)/ matches either the “puppy” or “puppies”
- It is equivalent to the union symbol ∪.

A simple example

- We want to find cases of the word “the”
  - /the/
- It misses the case then “the“ begins a sentence and is capitalized.
  - /[tT]he/
- It will incorrectly return cases with "the" embedded in other words (e.g., other or theology).
  - /\b[tT]he\b/
- We want to find “the” in cases where it might also have underlines or numbers nearby (the or the25).
  - /[^a-zA-Z][tT]he[^a-zA-Z]/
- It won’t find the word “the” when it begins a line.
  - /(^|[^a-zA-Z])[tT]he([^a-zA-Z]|$)/

Finite-state automata

- One can use a finite-state automaton to generate the strings of its language or to determine whether a given string indeed belongs to this language
- Finite-state automata are simply yet another way for defining (regular) languages,



FSA for the sheep language

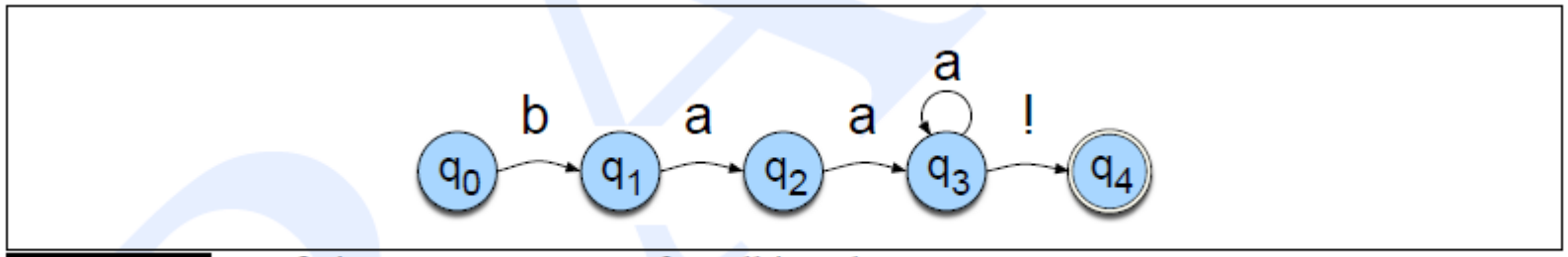


Figure 2.10 A finite-state automaton for talking sheep.

- More formally, a FSA is defined by five parameters:

$Q = q_0 q_1 q_2 \ldots q_{N-1}$	a finite set of N **states**
Σ	a finite **input alphabet** of symbols
q_0	the **start state**
F	the set of **final states**, $F \subseteq Q$
$\delta(q,i)$	the **transition function** or transition matrix between states. Given a state $q \in Q$ and an input symbol $i \in \Sigma$, $\delta(q,i)$ returns a new state $q' \in Q$. δ is thus a relation from $Q \times \Sigma$ to Q;

	Input		
State	b	a	!
0	1	Ø	Ø
1	Ø	2	Ø
2	Ø	3	Ø
3	Ø	3	4
4:	Ø	Ø	Ø

Non-Deterministic FSAs

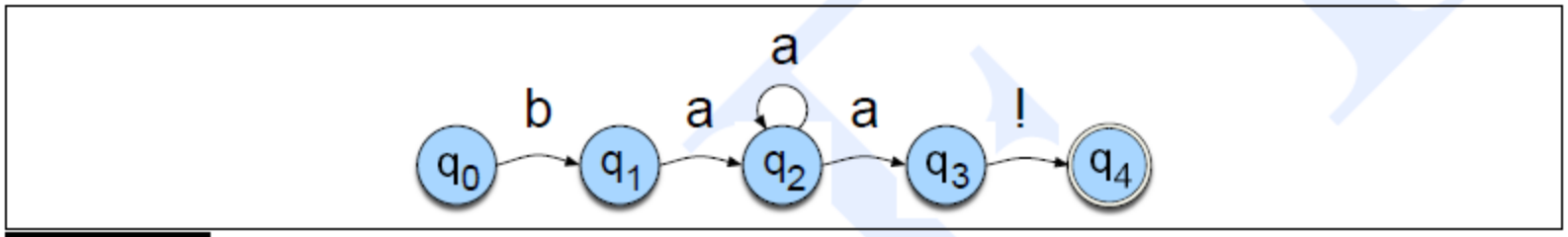


Figure 2.17 A non-deterministic finite-state automaton for talking sheep (NFSA #1). Compare with the deterministic automaton in Fig. 2.10.

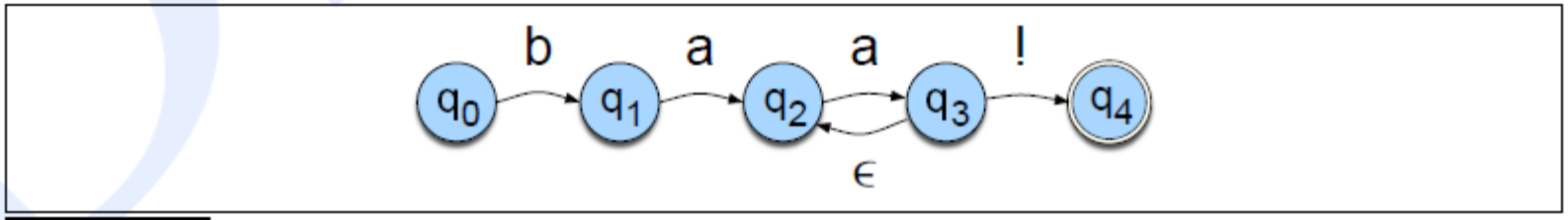


Figure 2.18 Another NFSA for the sheep language (NFSA #2). It differs from NFSA #1 in Fig. 2.17 in having an ϵ-transition.

Non-Deterministic FSAs

- For a Non-Deterministic FSA (**NFSA),** since there is more than one choice at some point, we might follow the wrong arc and reject a string when we should have accepted it.
- NFSAs need to perform search to make decision on acceptance.

Non-Deterministic FSAs – search

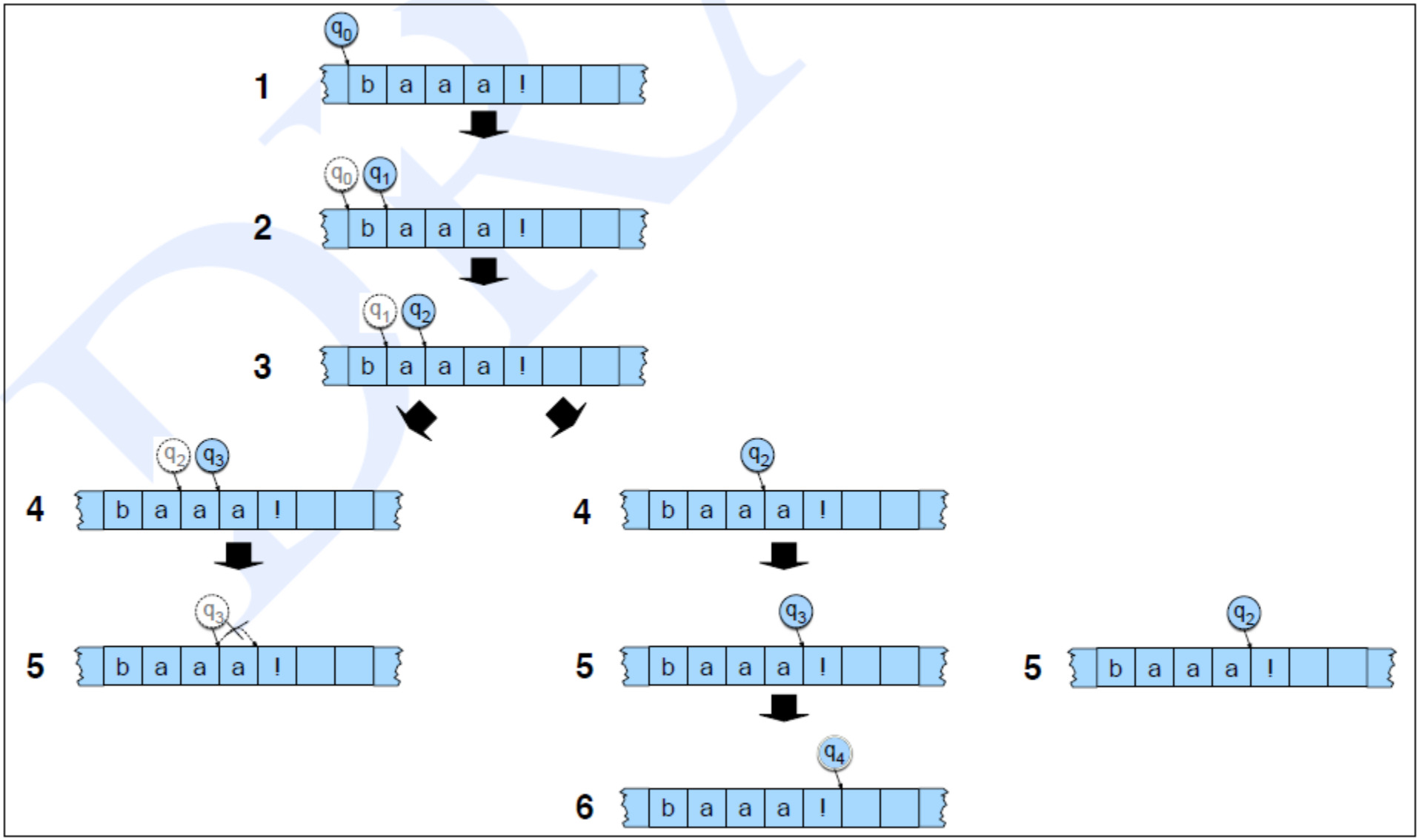


Figure 2.21 A breadth-first trace of FSA #1 on some sheeptalk.

Are NFSAs useful?

- Why do we need NFSA?
  - It is easier to construct a NFSA than a deterministic FSA (DFSA).
- Each NFSA can be converted to an equivalent DFSA.
  - However, the equivalent DFSA might be much larger.

Regular grammars

- Grammar is a set of rewriting rules for generating strings belonging to a language
- Regular grammars are equivalent to regular expressions.
- A given regular language can be characterized either by a regular expression or by a regular grammar.
- Regular grammars can either be right-linear or left-linear.

Symbols in rewriting rule

- The symbols that correspond to words/characters in the language are called **terminal**.
- The symbols that express generalizations of these are called **non-terminals**.

Right linear grammar

- Rules:
  - A single non-terminal on the left.
  - At most one non-terminal on the right-hand side.
  - If there is a non-terminal on the right-hand side, it must be the last symbol in the string

- $S \rightarrow aA$
- $S \rightarrow bB$
- $A \rightarrow aS$
- $B \rightarrow bbS$
- $S \rightarrow \epsilon$

Right linear grammar

- $S \rightarrow aA$
- $S \rightarrow bB$
- $A \rightarrow aS$
- $B \rightarrow bbS$
- $S \rightarrow \epsilon$

- $S \Rightarrow aA \Rightarrow aaS \Rightarrow aabB \Rightarrow aabbbS \Rightarrow aabbbaA$
  $\Rightarrow aabbbaaS \Rightarrow aabbbaa$

- Each time S expands, it produces either aaS or bbbS; thus this language corresponds to the regular expression $(aa \cup bbb)*$

Why study formal languages?

- Formal languages are not the same as natural languages.
- Could we use regular expressions to write a grammar for English?
- It tells us something about the formal properties of different aspects of natural language.
  - It would be nice to know where a language “keeps” its complexity;

How to tell if a language isn't regular?

- The Pumping Lemma (Please refer to the text book)
- $a^n b^n$ is not a regular language as it needs to keep the same number of a and b. However it is a context free language.

$$S \rightarrow a\ S\ b$$
$$S \rightarrow \epsilon$$

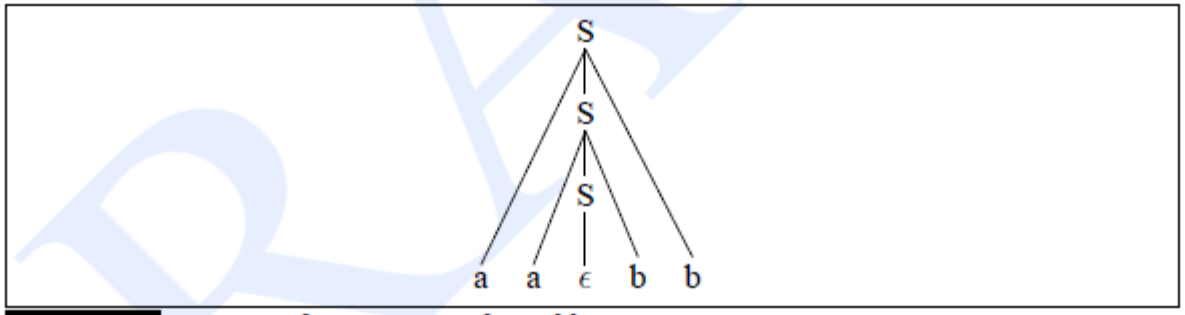


Figure 16.4 Context-free parse tree for *aabb*.

Is English a regular language?

- It is generally agreed that natural languages like English are not regular.
- Proof based on the **center embedded** structures
  - The cat likes tuna fish.
  - The cat the dog chased likes tuna fish.
  - The cat the dog the rat bit chased likes tuna fish.
  - The cat the dog the rat the elephant admired bit chased likes tuna fish.

  - (the + noun)^n (transitive verb)$^{n-1}$ likes tuna fish.
  - $L = x^n y^{n-1}$ *likes tuna fish*, $x \in A, y \in B$

Sublanguage

- Consider the following context-free language
  - L1 $= \{a^n b^n : n \in N\}$
- and L1 is contained in the regular language L:
  - L $= \{a^p b^q : p,q \in N\}$

- Thus, the fact that a language L contains a sublanguage that is very complex says nothing about the overall complexity of language L.
- Don't mix up with the **Chomsky hierarchy .**

Is natural language context-free?

- It is generally agreed that natural language is not context-free.
- $a^n b^m c^n d^m$ is not context free.
- Proof based on the **cross-serial dependencies.**

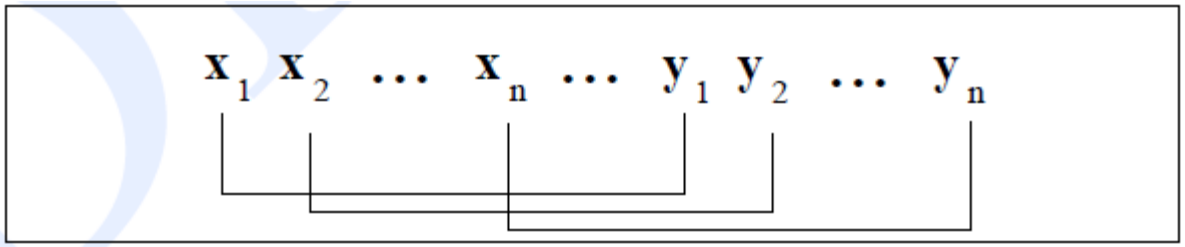
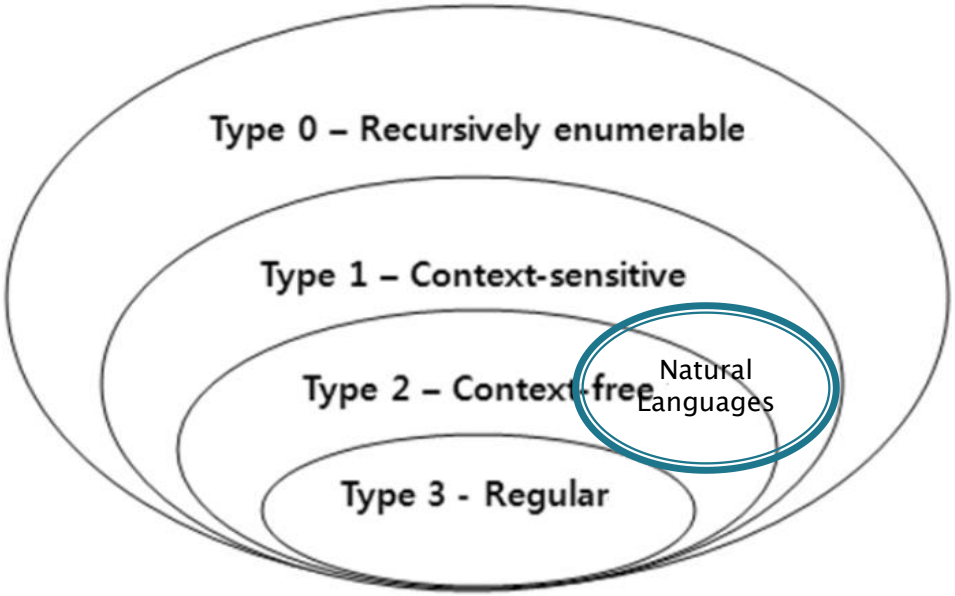


Figure 16.5 A schematic of a cross-serial dependency.

- Languages like Swiss German has this kind of property.

Where are natural languages

- An open problem over decades



Complexity and human processing

- Nesting or center-embedding causes difficulties when people try to read them.
  - “The cat the dog the rat the goat licked bit chased likes tuna fish.”
  - The cat [the dog [the rat [the goat licked] bit] chased] likes tuna fish.
  - “The pictures which the photographer who the professor met at the party took were damaged by the child.”
  - The pictures [which the photographer [who the professor met at the party] took] were damaged by the child.
  - Human memory limitation

Applying RE in our life

- Consider an intention classification task.
- A simple way to do text classification.
  - Different search pattern for each class
- 語音助手
  - 聽歌, 聽張學友的歌, 音樂, 想聽點的東西, 來首歌, 想點首那英的歌
  - 打電話, 打給某某某, 馬上撥給, 想找誰
  - 打開WIFI, 上不到網, 不想用4G
- 訂機票

Regular expression as a language

- Regular Expression itself is a language.
  - It is a meta-language: a language for denoting regular languages.
- Is the language to describe regular expression a regular language?
  - No, it is not regular. Consider that the language of balanced parenthesis alone is not even regular.

Natural languages vs. constructed languages

- A natural language is any language that has evolved naturally in humans through use without conscious planning or premeditation.

- A constructed language is a language which is consciously devised.
  - Computer programming languages can be perfectly processed by compilers.

Reading list

- Speech and Language Processing, version 2
  - Chapter 2 and 16
- Speech and Language Processing, version 3
  - Chapter 1