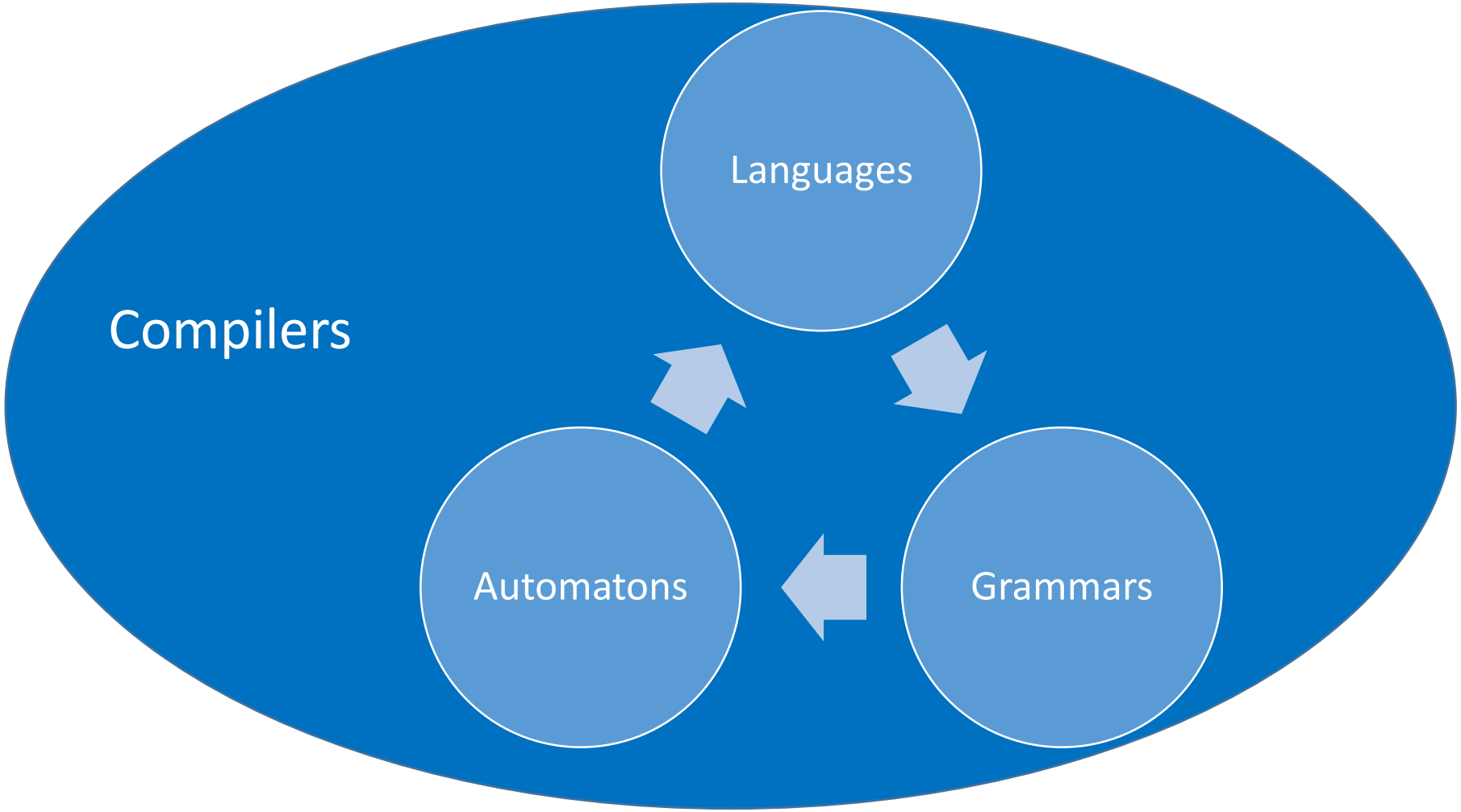
Một số khái niệm cơ bản của lý thuyết Automata và ngôn ngữ hình thức

Hà Chí Trung, BM KHMT, KHOA CNTT, HVKTQS

hct2009@yahoo.com

01685-582-102

Concepts



Languages

- English, Japanese, Vietnamese,…
- C, Python, C++, J++, JavaScript,…
- 1+1=? 1-2=? ☺
- 01, 01010, 101001011010101, ☹
- …☆®♥🂠♫♪☺☻☹☠☃🕱☢☯♨✉☊☭❀❀✿✾✻✢✵✡★✰✰❉❄✪✞☪☮☮✠☣♚♔♛♕♜♖♞♘♝♗🕴🕇웃유…
- …

Grammars

<câu>→<chủngữ><vịngữ>

<chủngữ>→tôi | anh | chị

<vịngữ>→<độngtừ><danhtừ>

<độngtừ>→ăn | uống

<danhtừ>→cơm | phở | sữa|…

Grammars

$$G = < \Sigma, \Delta, S, P >,$$

where:

- Σ - **terminal symbol**;
- Δ , $\Delta \cap \Sigma = \emptyset$, – **nonterminal symbol**;
- $S \in \Delta$ - **start variable**;
- P - **production rules** $P = \{\alpha \rightarrow \beta \mid \alpha, \beta \in (\Sigma \cup \Delta)^*\}$.

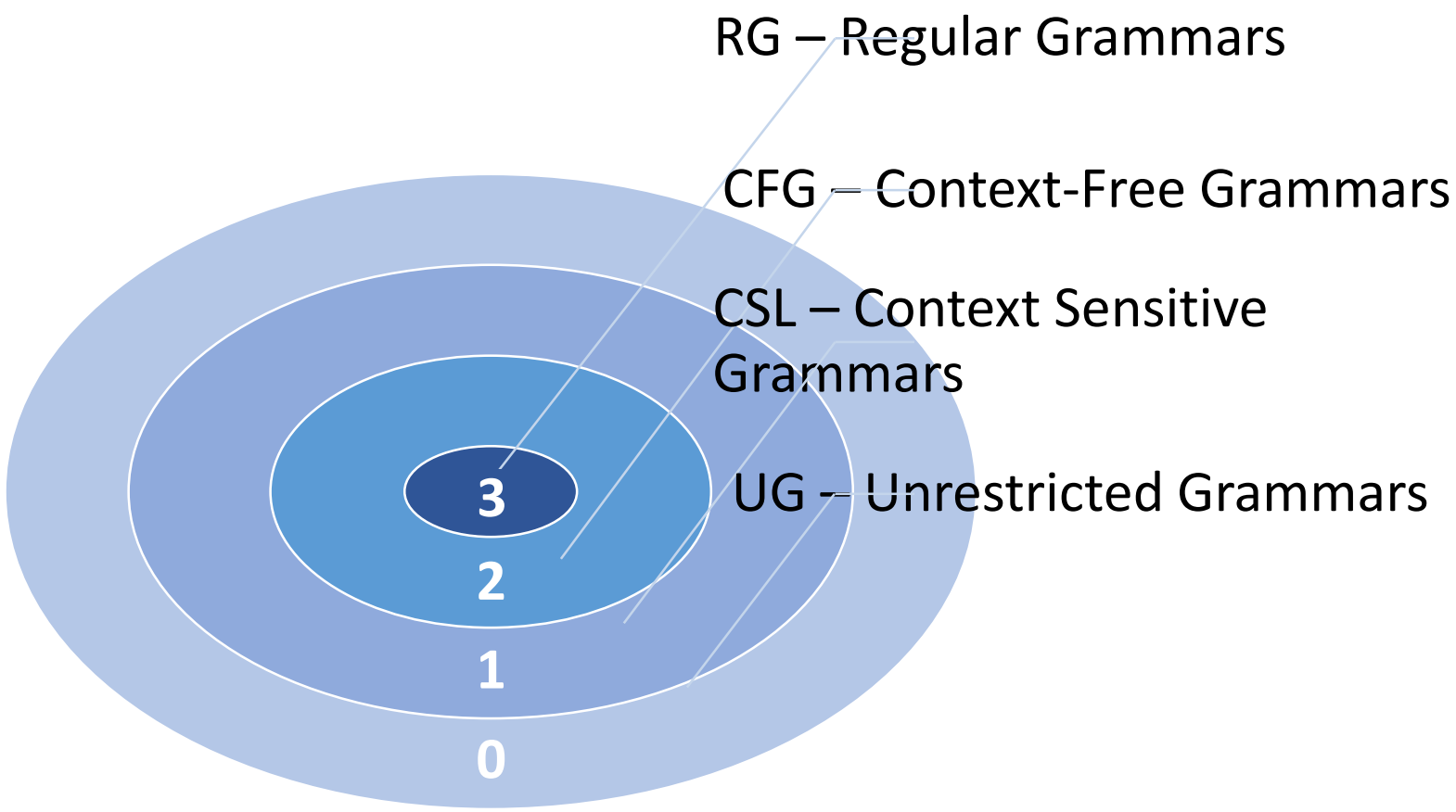
Grammars – Backus Naur Form (BNF)

- **BNF for Java**

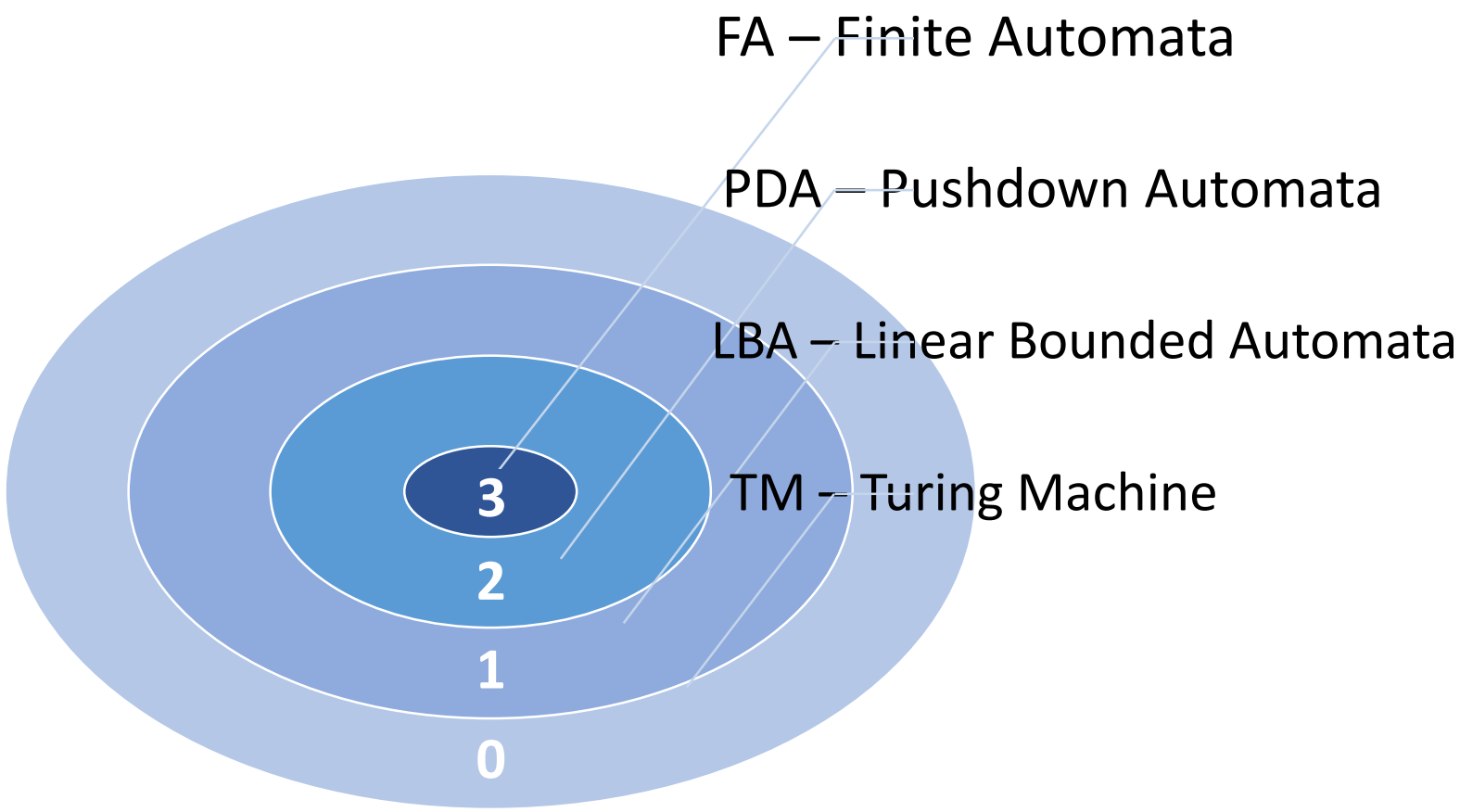
  http://cui.unige.ch/db-research/Enseignement/analyseinfo/JAVA/BNFindex.html

  expression ::= numeric_expression | testing_expression | logical_expression | string_expression | bit_expression | casting_expression | creating_expression | literal_expression | **"null"** | **"super"** | **"this"** | identifier | (**"("** expression **")"**) | (expression ((**"("** [arglist] **")"**) | (**"["** expression **"]"**) | (**"."** expression) | (**","** expression) | (**"instanceof"** (class_name | interface_name))))

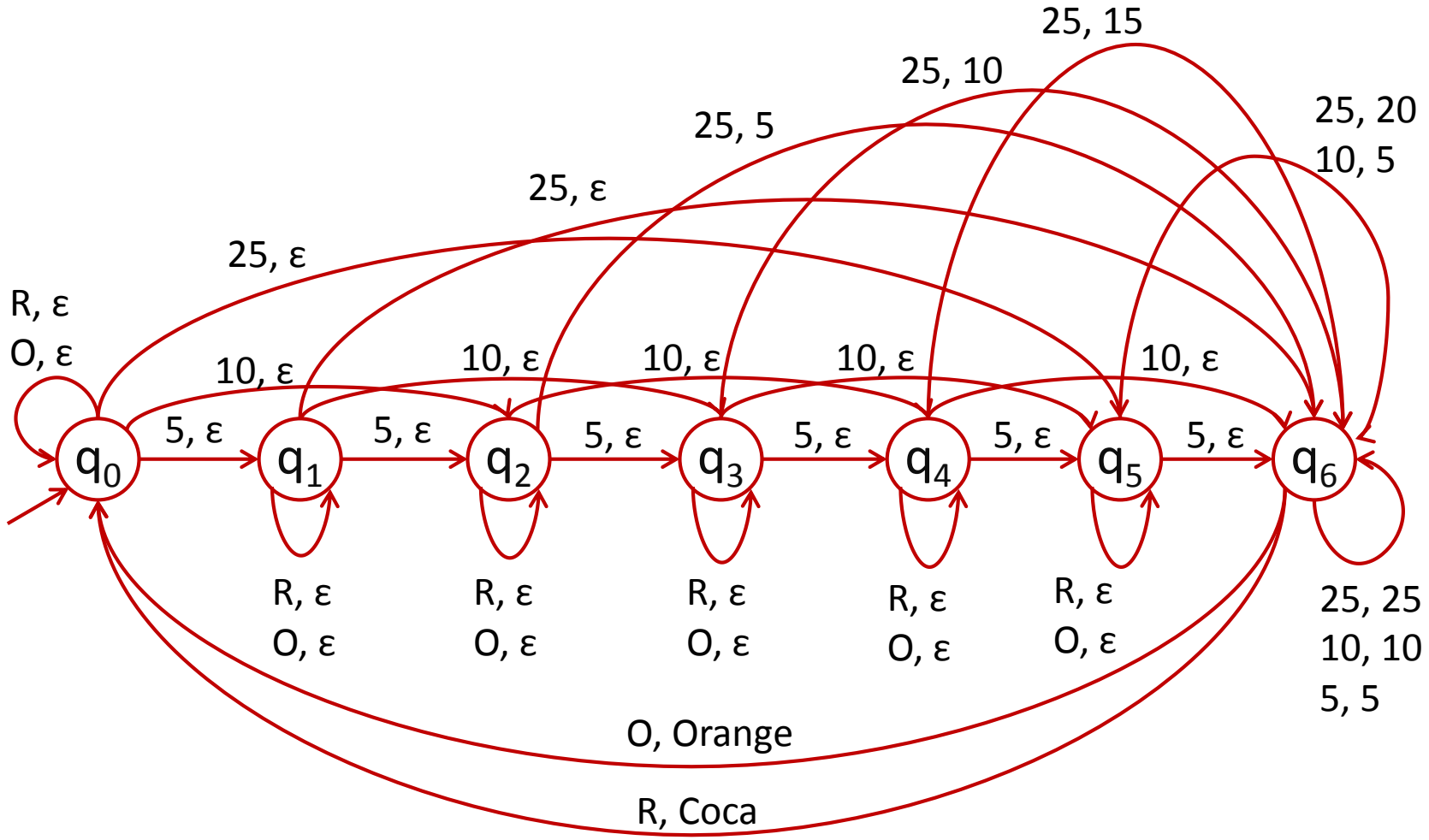
Grammars – Chomsky (1956)



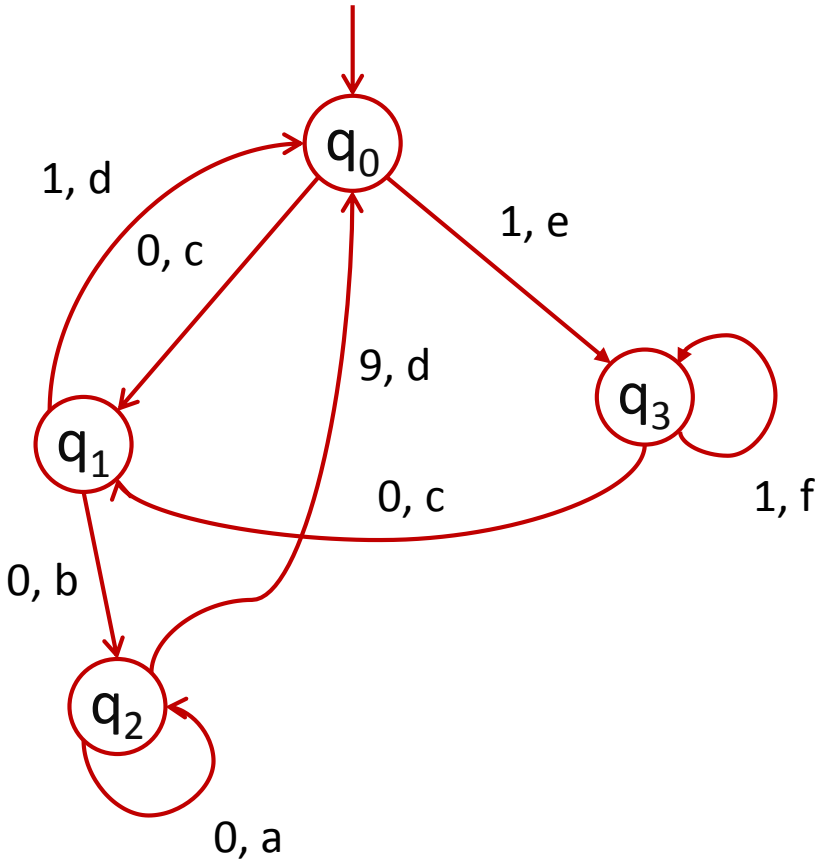
Automata


FA – Finite Automata
PDA – Pushdown Automata
LBA – Linear Bounded Automata
TM – Turing Machine
3
2
1
0

Kenneth H. Rosen (Ch.13)



Automata



- **An đi học…**

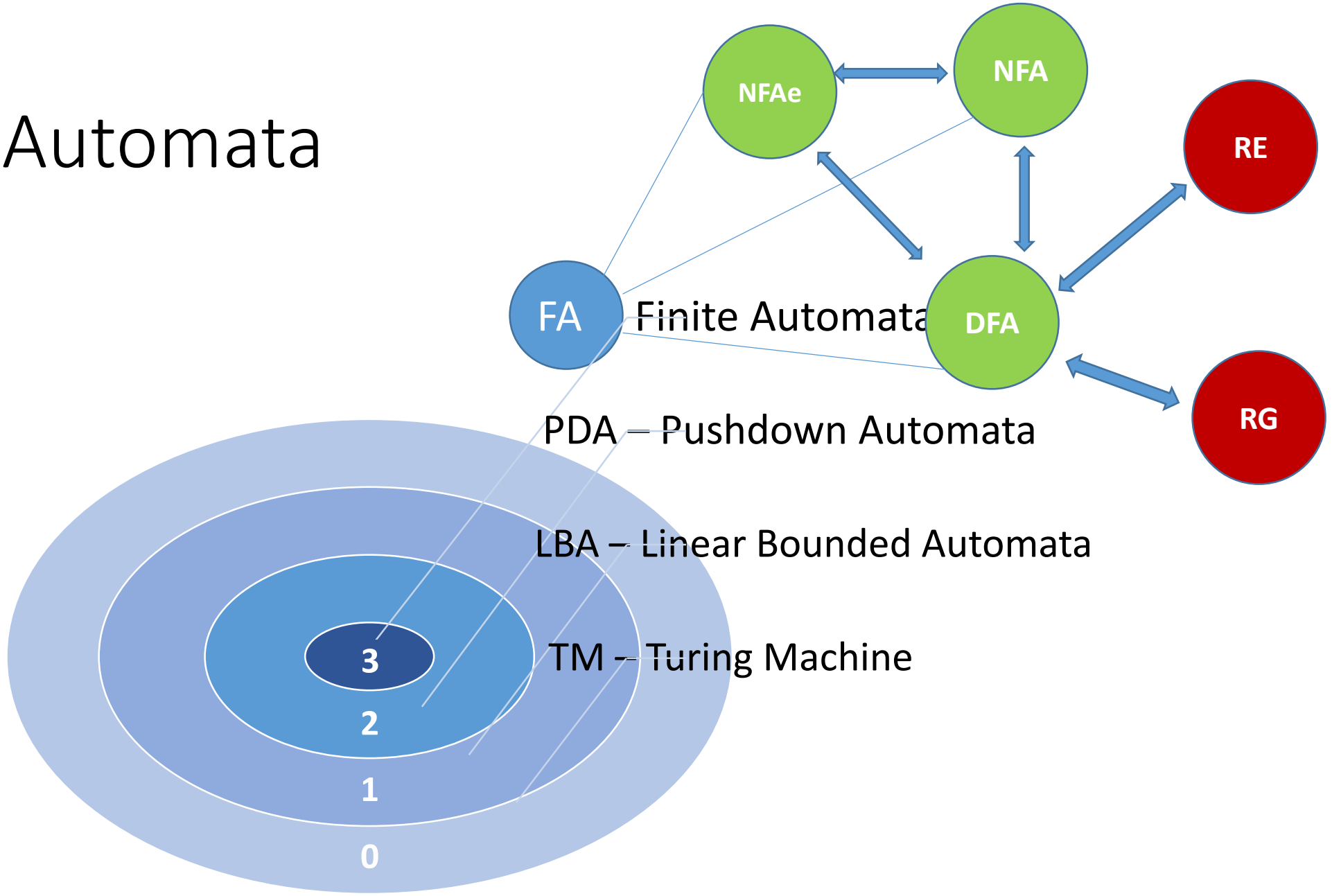
  01011000…

  01110…

- ***Bố An (Bình):***

a = ”cầm roi”; b= “mắng chửi”; c = “dỗ dành”; d = “hy vọng”; e = “vui sướng”; f = “khen ngợi”.

Finite Automata



Finite Automata

- **NFAε:** Nondeterministic Finite Automata with ε-transitions
- **NFA:** Nondeterministic Finite Automata
- **DFA:** Deterministic Finite Automata
- **RE:** Regular Expression
- **RG**: Regular Grammar
- …

Finite Automata

DFA:

$$A=<Q, \Sigma, \delta, q_0, F>,$$

where:

Q : **states** (p, q…);

Σ : finite set of *symbols*, called the *alphabet* (a, b, c …);

δ : $Q \times \Sigma \rightarrow Q$, **transition function,**

$\delta(p, a) = q$ or $\delta(p, a) = \emptyset$, $p, q \in Q$, $a \in \Sigma$;

$\mathbf{q_0} \in Q$: **start state**;

$\mathbf{F} \subseteq Q$: **finish (accept) states**.

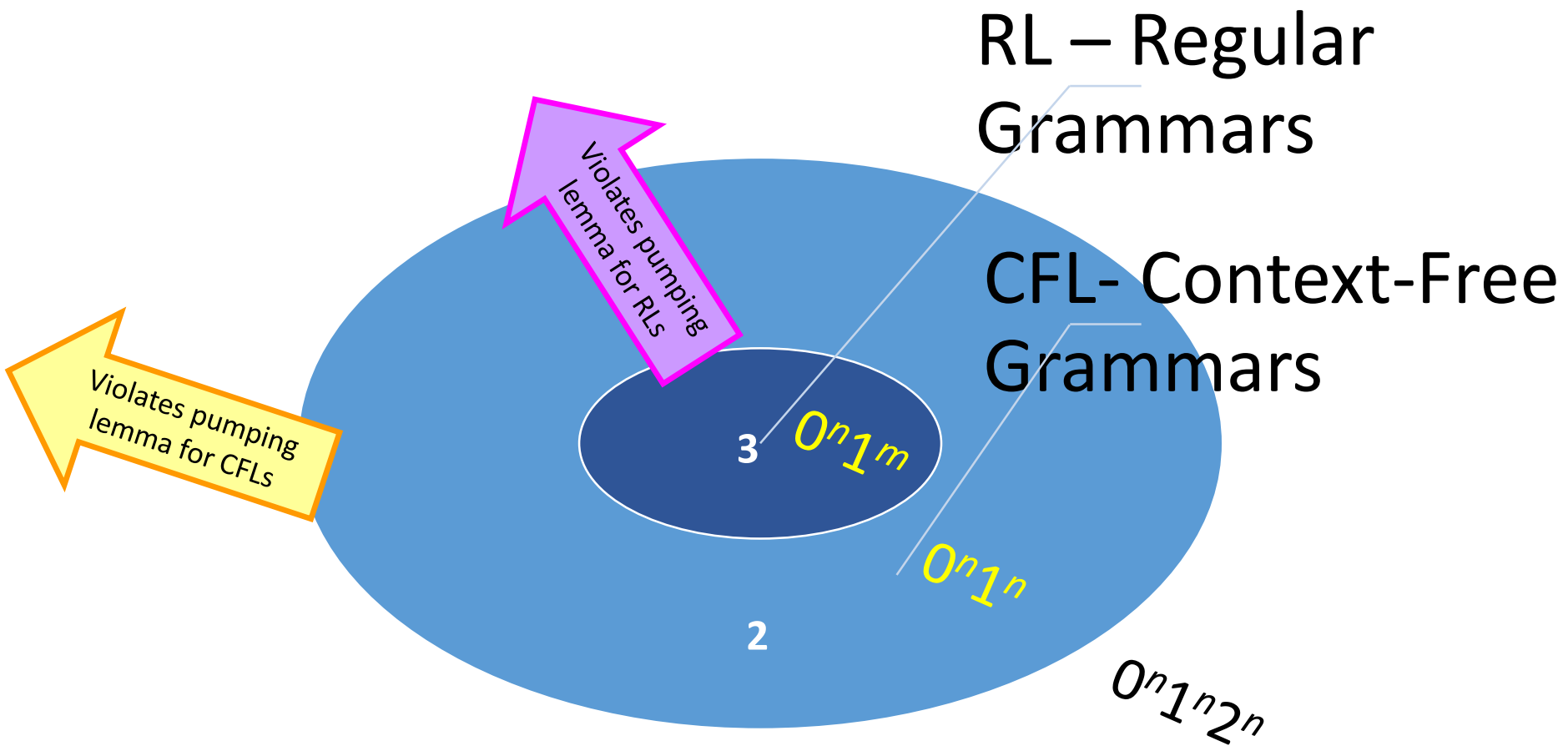
NFA: **δ**: $Q \times \Sigma \rightarrow 2^Q$

NFAε: **δ**: $Q \times (\Sigma \cup \{\varepsilon\}) \rightarrow 2^Q$

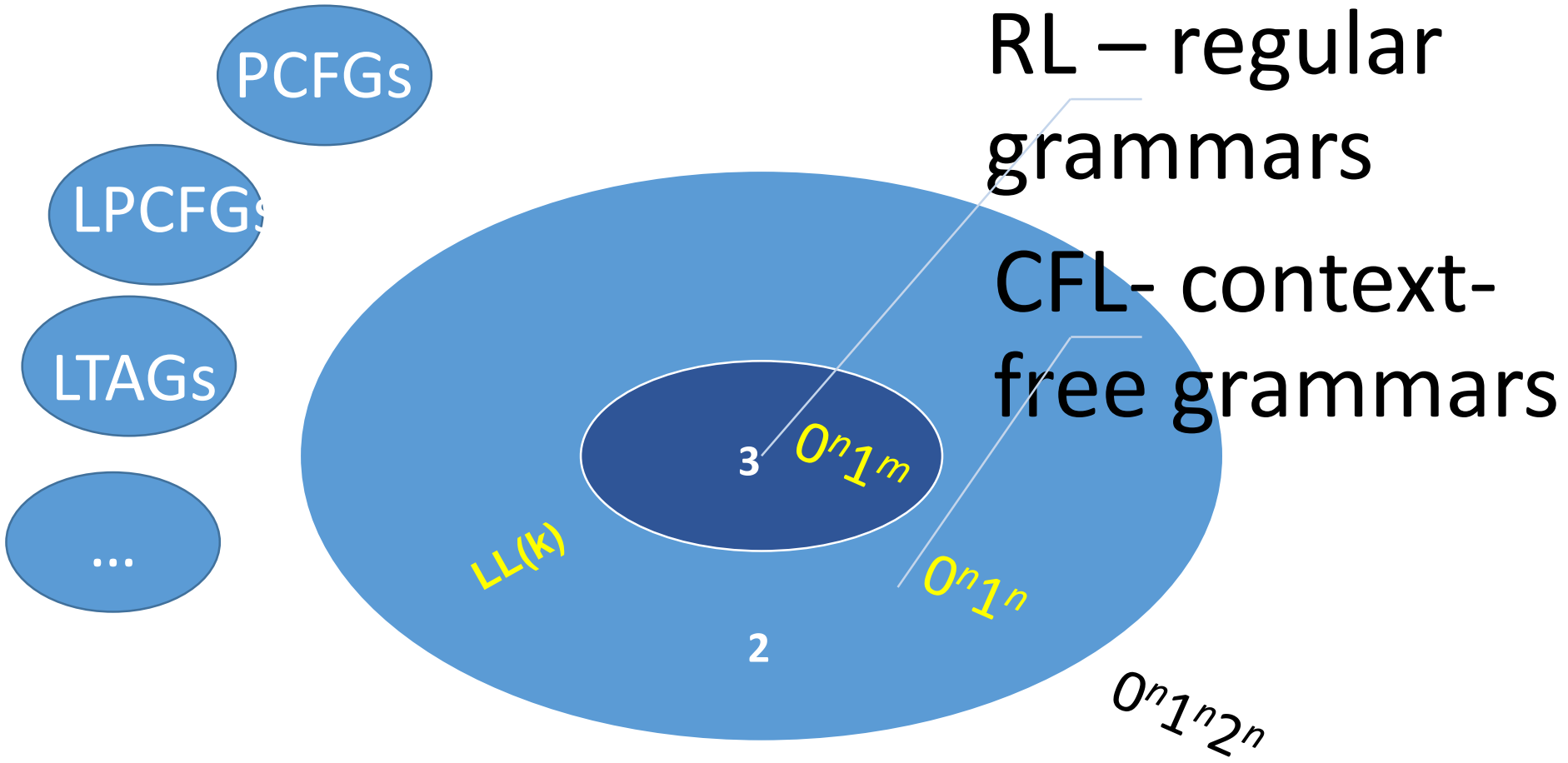
Regular Expressions

- 0*1*01
- (0+1)*01
- .*\.txt$
- \b[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,4}\b
- hct2009@yahoo.com
- ^[A-Z0-9._%+-]+@[A-Z0-9.-]+\.(?:[A-Z]{2}|com|org|net|edu|gov|mil|biz|info|mobi|name|aero|asia|jobs|museum)$
- …

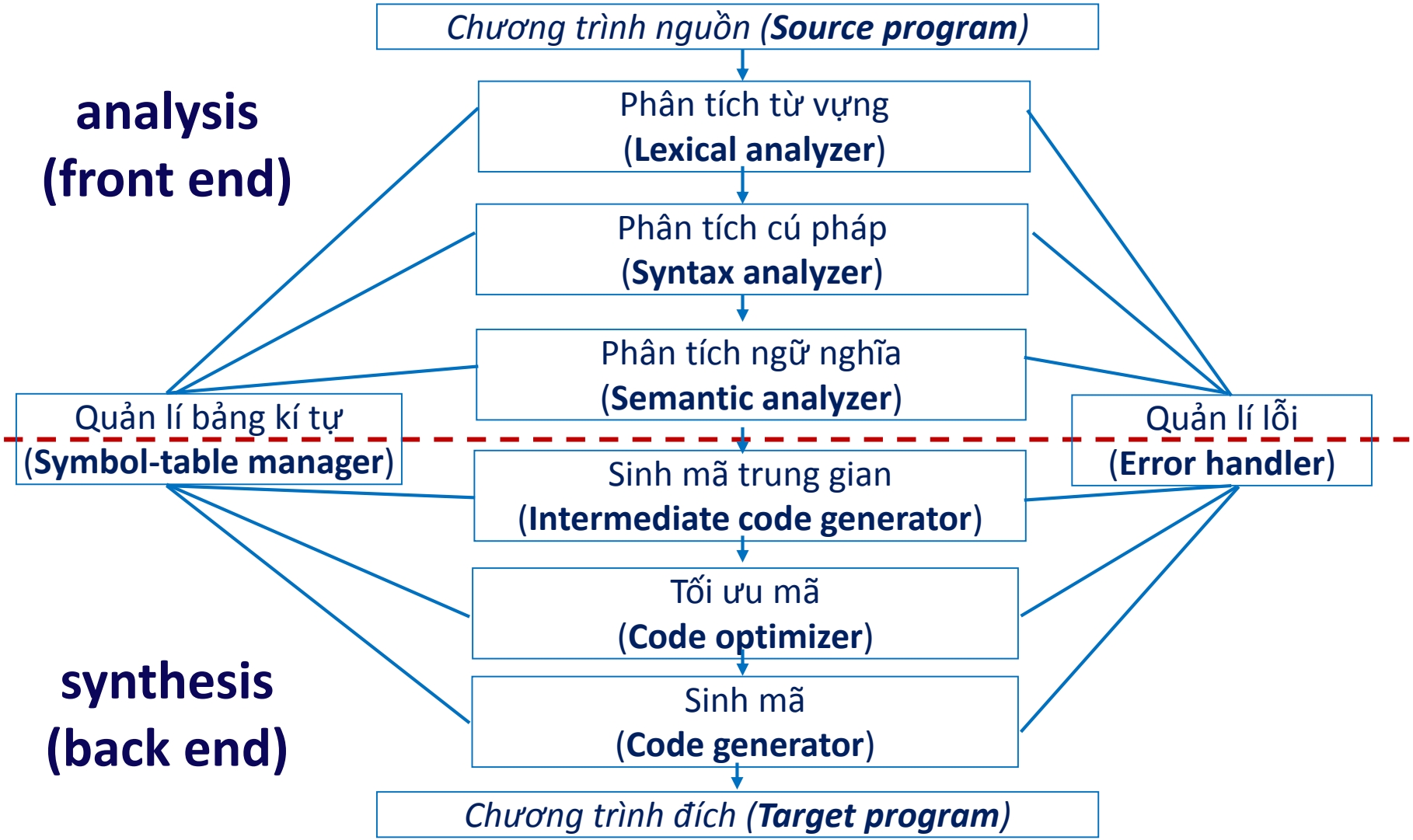
Grammars(ct.)


RL – Regular Grammars
CFL- Context-Free Grammars
Violates pumping lemma for RLs
Violates pumping lemma for CFLs
3
0^n1^m
0^n1^n
2
$0^n1^n2^n$

Grammars(ct.)

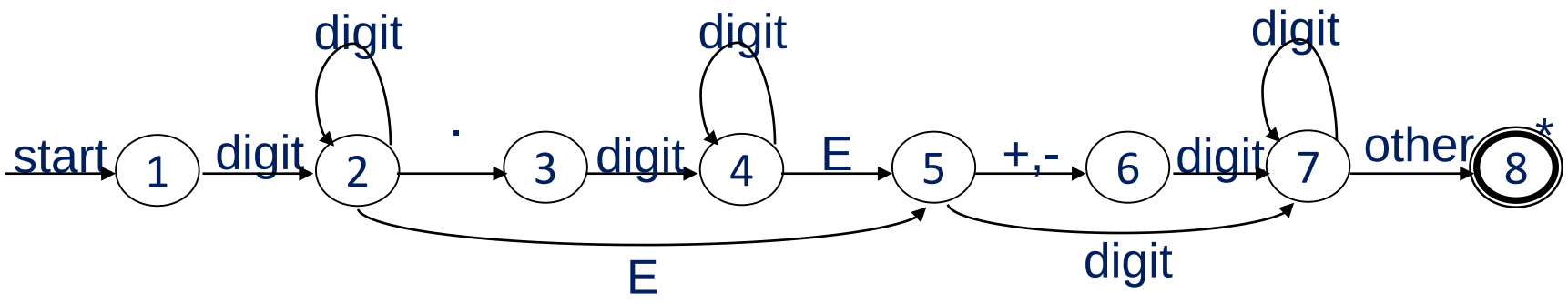

PCFGs
LPCFGs
LTAGs
...
RL – regular grammars
CFL- context-free grammars
3
0^n1^m
LL(k)
0^n1^n
2
$0^n1^n2^n$

Compilers

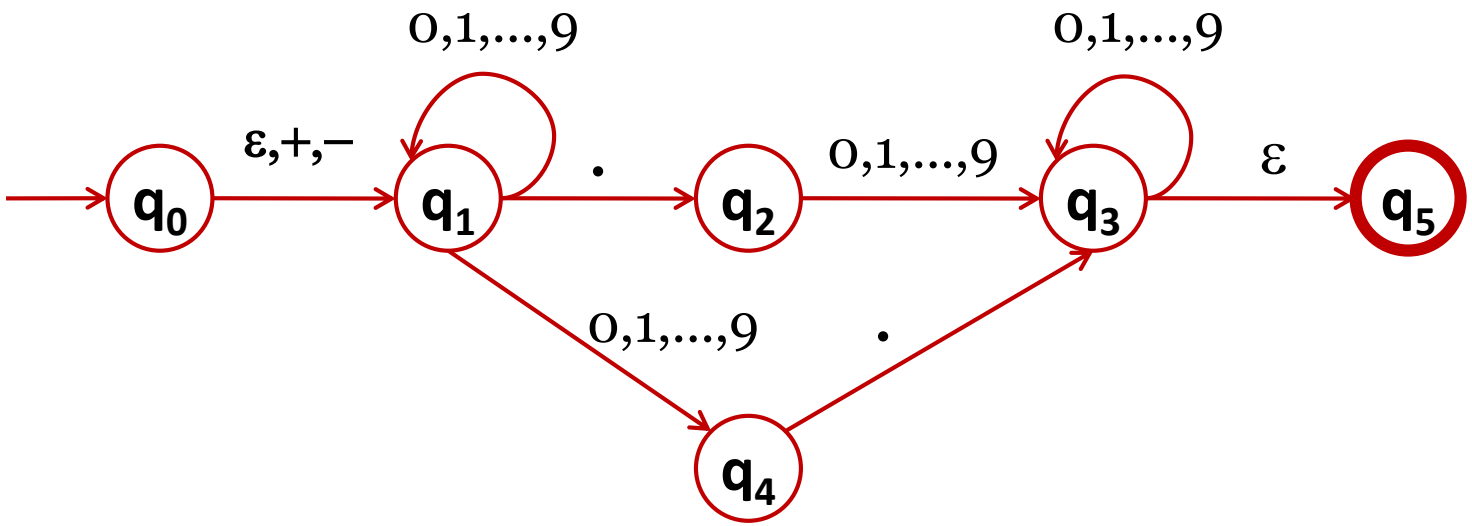

Chương trình nguồn (**Source program**)
analysis
(front end)
Phân tích từ vựng
(**Lexical analyzer**)
Phân tích cú pháp
(**Syntax analyzer**)
Phân tích ngữ nghĩa
(**Semantic analyzer**)
Quản lí bảng kí tự
(**Symbol-table manager**)
Quản lí lỗi
(**Error handler**)
Sinh mã trung gian
(**Intermediate code generator**)
Tối ưu mã
(**Code optimizer**)
synthesis
(back end)
Sinh mã
(**Code generator**)
Chương trình đích (**Target program**)

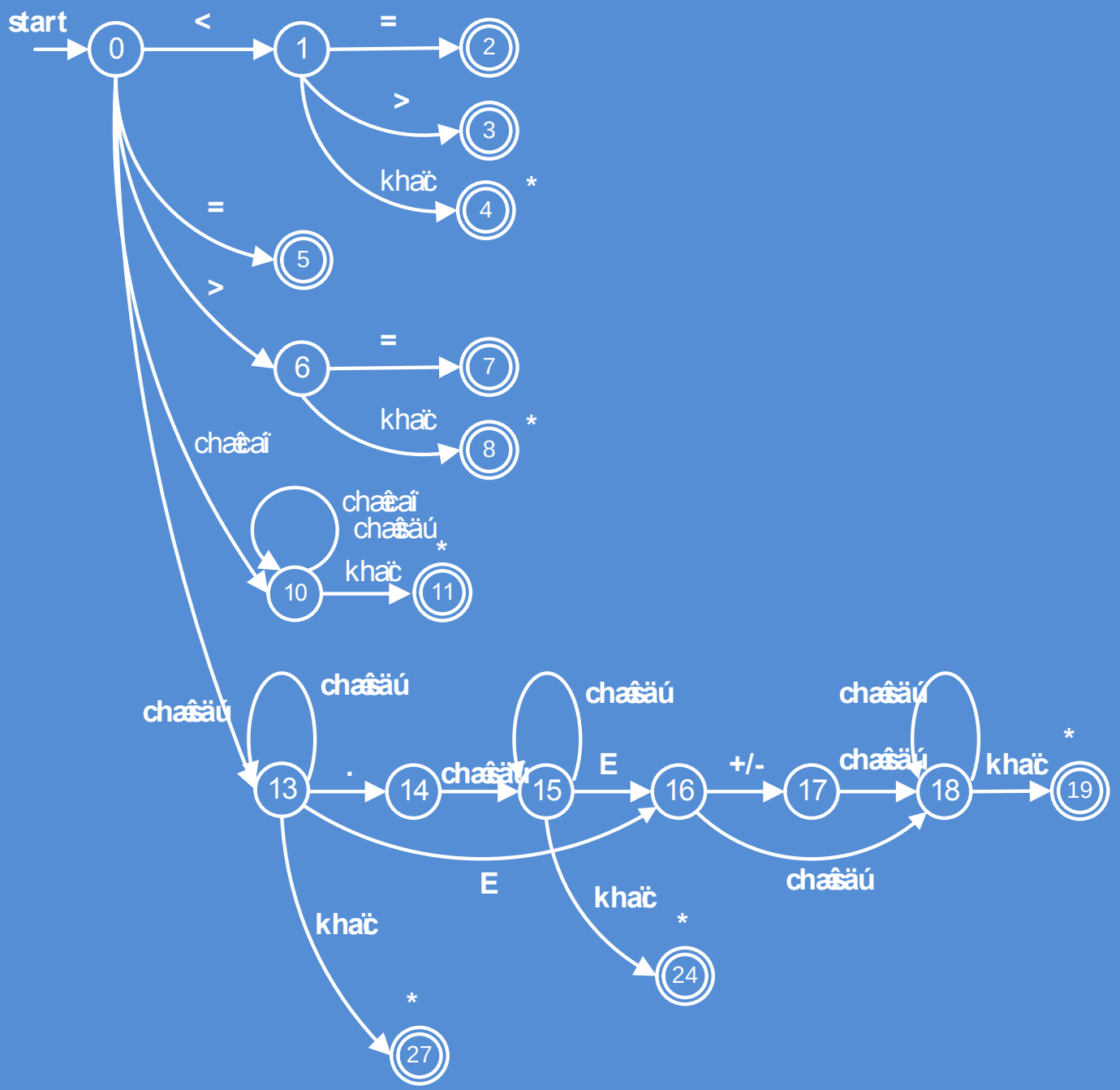
Compilers

- **Pascal:**



- **C, C++:**




start
0
<
1
=
2
>
3
khaùc
4
*
=
5
>
6
=
7
khaùc
8
*
chaøcaïi
chaøcaïi
chaøsäú
10
khaùc
11
*
chaøsäú
chaøsäú
chaøsäú
chaøsäú
chaøsäú
13
.
14
chaøsäú
15
E
16
+/-
17
chaøsäú
18
khaùc
19
*
E
khaùc
*
24
khaùc
*
27

Parsing

- Top-down parsing.
- Bottom-up parsing.
  - Phân tích đệ quy ($O(c^n)$);
  - thuật toán phân tích **CYK** (**Coke-Younger-Kasami**) ($O(n^3)$);
  - (thuật toán phân tích **Earley)** ($O(n^3)$ hoặc $O(n^2)$ hoặc $O(n)$);
  - **LL(k)** for top-down parsing ($O(n)$);
  - **LR(k)** for với bottom-up parsing ($O(n)$).

Parsing

- LL(k) (Left-to-right parse, Leftmost-derivation, k-symbol lockahead);
- PCFGs
- LPCFGs
- LTAGs