BioInformatics

Automata — Formal Languages — Complexity

J.-M. Steyaert

École Polytechnique

How to model biological features ?

Equations, Probabilistic models, Statistics, Networks, etc.

Stringology and algorithms : extract information from genome
Find Patterns in Strings (nucleotides, amino-acids) : local, long distance, embedded, global,...
Compute probability distributions, limiting and tail distributions, etc.
Recombine and predict tentative structures

Formal language theory : framework for expressing models

Automata, Algorithms (e.g. dynamic programming, exhaustive search, heuristic search) : tools to find pattern occurrences

A brief overview of basic tools : Finite Automata, Context Free Languages, Others...

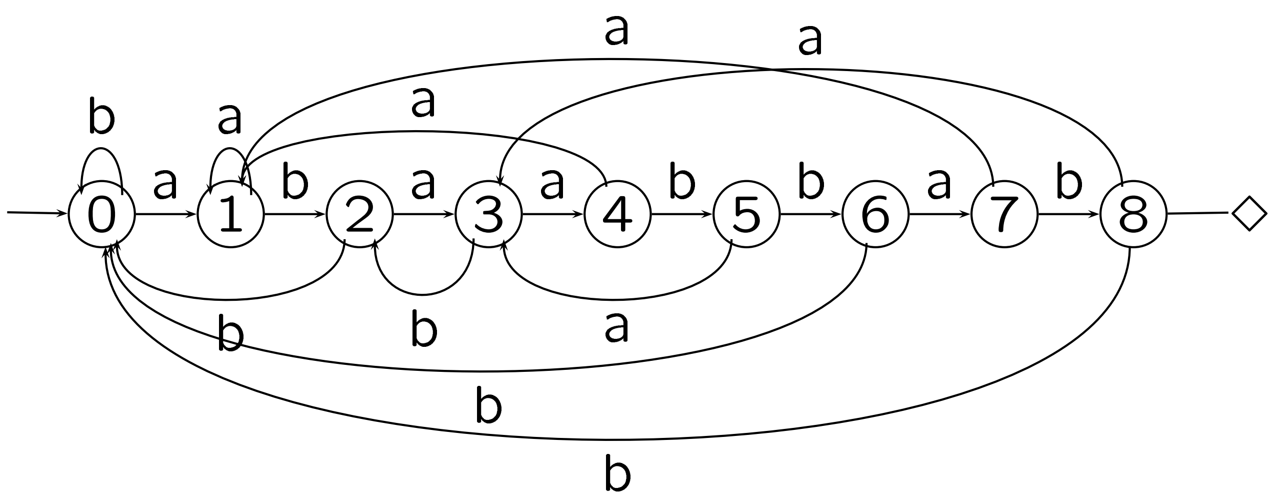
1. Finite automata and regular expressions : Kleene's theorem
2. Determinism vs. nondeterminism ; Expressive power
3. Context free grammars and languages
4. Pushdown stack automata
5. The CKY algorithm : dynamic programming
6. Other classes : NP-problems
7. Long range patterns and biology

Finite Automata : Definitions

- Alphabet, word : A finite set, $w \in A^k$ word of length k
- Monoid A^* (free) : words of any length
- Empty word : ε has length 0
- Concatenation : $f.g$; $w.\varepsilon \;=\; \varepsilon.w \;=\; w$, neutral element

Automaton : $\mathcal{A} = \langle A, Q, q_0, F, \Delta \rangle$,
alphabet : A, states : Q, initial state : q_0, final states : F
transition table (function) : $\Delta \;:\; Q \times A \longrightarrow Q$...
will be strengthened ...

Filters and pattern recognition



The minimal automaton for pattern $m = abaabbab$

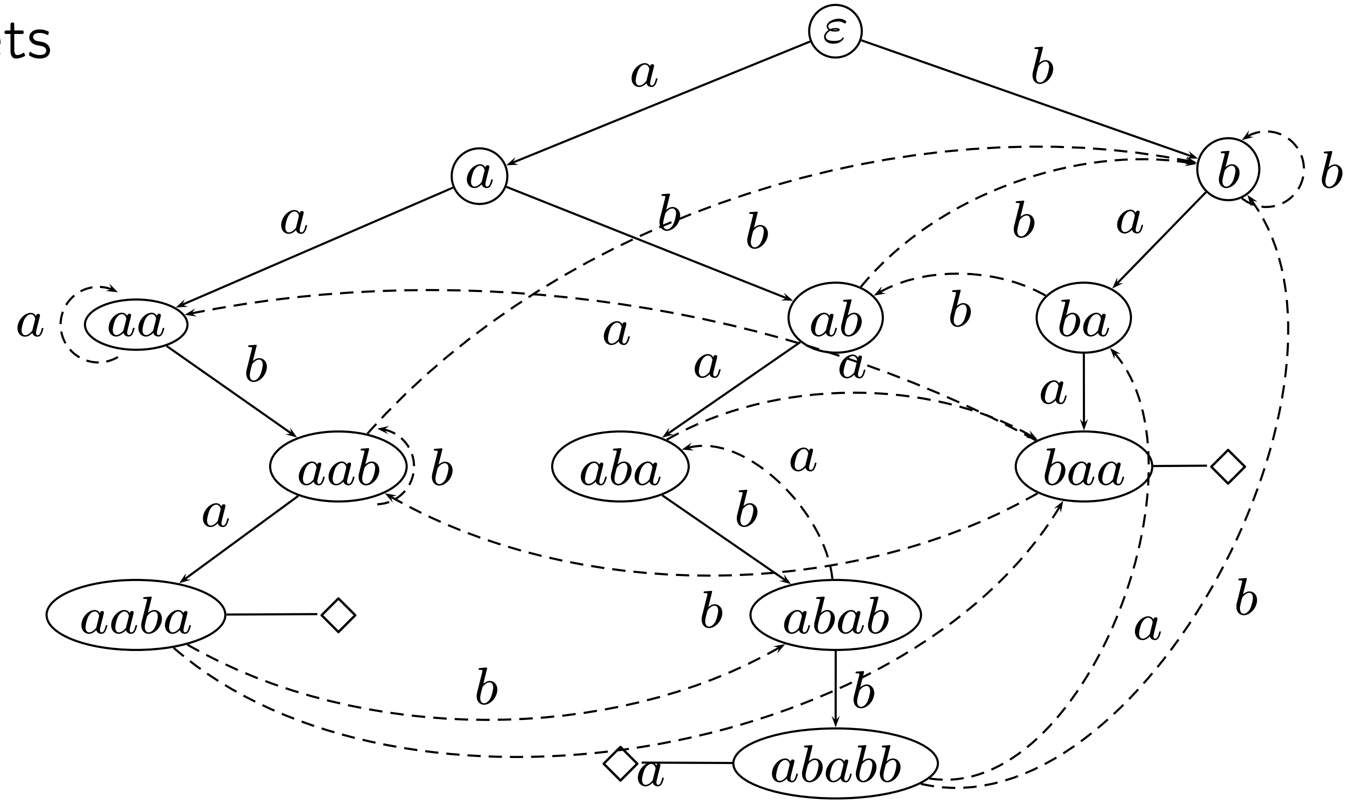
How does it work ?

Configuration : (w, q), w word (to be read), q current state

Computation of automaton $\mathcal{A}$ on w : any sequence of configurations $c_0, c_2, ..., c_k$, s.t. one can go from c_i to c_{i+1} via a transition $q_i, a_{i+1}, q_{i+1}) \in \mathcal{A}$, and $w_i = a_{i+1}w_{i+1}$ for $0 \leq i < k$.

Acceptation : Computation starts with $c_0 = (w, q_0)$, ends with $c_k = (\varepsilon, q)$, for $q \in F$.

Pattern sets



Automaton for pattern set $M = \{ababb, aaba, baa\}$; the automaton skeleton is in plain arrows

Technicalities : Set of words leading to q : $\mathcal{L}_q$.
Recursive (simultaneous) definition :
– $\varepsilon \in \mathcal{L}_{q_0}$ (Empty word allows access to initial state)
– $\mathcal{L}_q = \{aw \mid a \in A, q' \in Q, w \in \mathcal{L}_{q'}, (q', a, q) \in \Delta\}$ (transirions)
– $\mathcal{L}(\mathcal{A}) = \cup_{q \in F} \mathcal{L}_q$ (last state is final) : Language recognized by $\mathcal{A}$.

Regular Expressions :
– finite set A : alphabet
– binary operators « $+$ » (union), « . » (concatenation),
unary« $*$ » (quasi-inverse)

Formulae $\mathcal{R}eg$: – any element of A is in $\mathcal{R}eg$;
– for any $e_1, e_2 \in \mathcal{R}eg$, then $e_1 + e_2 \in \mathcal{R}eg$ and $e_1.e_2 \in \mathcal{R}eg$;
– for all $e \in \mathcal{R}eg$, then $(e) \in \mathcal{R}eg$ and $(e)^* \in \mathcal{R}eg$.

Sets defined by regular expressions

$\mathcal{L}(e) \subset A^*$ language associated to e :

- $\mathcal{L}(\varepsilon) = \{\varepsilon\}$;
- $\mathcal{L}(a) = \{a\}$, for any $a \in A$;
- $\mathcal{L}(e_1.e_2) = \{w_1.w_2 \mid w_1 \in \mathcal{L}(e_1) \ \& \ w_2 \in \mathcal{L}(e_2)\}$;
- $\mathcal{L}(e_1 + e_2) = \{w \mid w \in \mathcal{L}(e_1) \ \vee \ w \in \mathcal{L}(e_2)\}$;
- $\mathcal{L}(e^*) = \{\varepsilon\} \cup \mathcal{L}(e) \cup (\mathcal{L}(e))^2 \cup (\mathcal{L}(e))^3 \cup ...$

Examples : any kind of local or repeated patterns

From regular expressions to finite automata : Non-determinism ! ! !

Automata compiling :
– finite sets
– concatenation
– union ! !
– quasi-inverse ! !

Non-deterministic finite automata
Transition table : $\mathbf{\Delta}\ :\ Q \times A \longrightarrow Q$ becomes

$$\mathbf{\Delta} \subset Q \times (A \cup \{\varepsilon\}) \times Q$$

A word is accepted **iff** one computation ends in a final state

Theorems

1.To any regular expression, one associate effectively (by an algorithm) a non deterministic finite automaton which recognizes the language defined by the expression.

$$\mathcal{R}at = \mathcal{L}(\mathcal{R}eg) \subset \mathcal{A}ut.$$

2. The class $\mathcal{A}ut$ contains the finite sets and is closed by :
– union, product, Kleene's star
– intersection, morphism, complementation.

3. Reciprocal : Kleene's theorem
All the languages $\mathcal{L}_q$ (set of words that mark a path from the initial state to state q) can be represented by regular expressions.

Theorem : Any non deterministic finite automaton can be transformed into an equivalent deterministic automaton.

Price to be payed : Exponential increase in the number of states !

Proof : By simulation and/or by generic construction

The exponential gap can be achieved (especially in biology !) Big problem in combinatorics and statistics...

To be or not to be in $\mathcal{R}at$? The iteration lemma

The automaton is unable to memorize more features than its number of states !

Thm : Let L be a rational language ; there exists an integer C such that any word f in L longer than C can be written $f = uxv$, ($|x| > 0$), and then $ux^*v \subset L$.

Morality : Finite automata (even probabilistic) are adapted to local or repetitive patterns.

Facts

Languages $P_1 = \{a^n b^n \mid n \geq 0\}$ and $P_3 = \{w \in (a+b)^* \mid |w|_a = |w|_b\}$ are NOT in $\mathcal{R}at$.

— $P_3 = \{ab^{i_1}ab^{i_2}a...ab^{i_n}a \mid n \geq 0 \ \& \ \exists k \ k = i_k\}$?

— $P_4 = \{ab^k ab^{k^2}\}$?

— $P_5 = \{ww^R \mid w \in (a+b)^*\}$? —> Perec et al.

How can we describe the walks along a treelike wall ? a maze ? Circuits in a Eulerian graph ? Moves of a train and its carriages on a T-shape railway ? Secondary structures of t-RNAs ?

Context-free Grammars

Definition :

A CF grammar (Chomsky) is a rewriting system defined by :

— V, finite set of variables ;

— T, finite set of terminals (language alphabet) ;

— $\mathcal{R} \subset V \times (V \cup T)^*$, finite set of rewriting rules ;

— $S \in V$, axiom symbol.

Derivation

Two words $w, w' \in (V \cup T)^*$; w rewites in w' (or w' derives in one step from w), iff :

— w writes $w = xvy$, $x, y \in (V \cup T)^*$ and $v \in V$;

— w' writes $w' = xfy$, $x, y \in (V \cup T)^*$ (the same) and $f \in (V \cup T)^*$;

— (v, f) is a rule in $\mathcal{G}$.

Sequence of rewritings

$$w \rightarrow w_1 \rightarrow w_2 \rightarrow \ \ldots \ \rightarrow w_n = w'$$

is called derivation (noted $w \xrightarrow{*} w'$).
Furthermore, if w' is on the terminal alphabet ($w' \in T^*$), the derivation is terminal.

A word can be derived in several ways.

Algebraic Expressions

```
Expression ::= Term | Term + Expression
Term       ::= Factor | Factor Terme
Factor    ::= ( Expression ) | Var | Num
Var        ::= a | b | ... | z
Num        ::= 0 | 1 | ... | 9
(* Variables and Integers write with one symbol *)
```

Sketch of a grammar for algebraic expressions

Derivations do not depend on the context and are usually represented by trees

The family $\mathcal{A}lg$

(These languages are solutions of polynomial non-commutative systems)

Thm : *The class $\mathcal{A}lg$ is closed by union, product and quasi-inverse. It is closed by homomorphism and algebraic substitution*

Thm : *The class $\mathcal{A}lg$ is closed by intersection with a rational language*

It is NOT closed by Intersection nor by Complementation

How to recognize Context-Free languages ?

Answer : With a Pushdown-stack automaton

Example : Palindromic words

The automaton is not necessarily deterministic !

Thm : The two classes are identical

Example of a pushdown stack automaton :

$T = \{a, b\}, Q = \{0, 1, 2\}, P = \{b, c\}, q_0 = 0, p_0 = c$

$\Delta = \{(0, a, c) \rightarrow (0, c), (0, b, c) \rightarrow (0, bc), (0, b, b) \rightarrow (0, bb)$, (push on stack the b's)

$(0, b, b) \rightarrow (1, bb)$ (change mode)

$(1, a, b) \rightarrow (1, \varepsilon)$ (count as many a's as the $\#b$)

$(1, b, c) \rightarrow (2, c)$ (pattern is found)

$(2, a, c) \rightarrow (2, c), (2, b, c) \rightarrow (2, c), (2, b, c) \rightarrow (2, \varepsilon)\}$ (read the tail).

This automaton recognizes words $a^* b a^{i_1} b a^{i_2} b ... b a^{i_k} b$ s.t. there exists a j for which $j = i_j$.

Deterministic parsing (recognition) of CF languages (in Chomsky normal form)

Chomsky normal form : A=a or A=BC for all the rules (binary derivation trees)

Convention :
– variables $A, B, C, ... \in V$
– words on the terminal alphabet $(f, g, h, ... \in T^* \backslash \{\varepsilon\})$

Function Drv : $V \times T^* \rightarrow \{0, 1\}$ (boolean values)

$$
\begin{array}{lll}
Drv(A, f) = & & \{\text{for } f \neq \varepsilon\} \\
& \text{if } f \in T & \text{then } (A = f) \in \mathcal{R}? \\
& \text{else} & \exists\ g, h \neq \varepsilon,\ \exists\ A, B, C \\
& & \{\ f = gh\ \&\ A = BC \in \mathcal{R}\ \&\ Drv(B, g)\ \&\ Drv(C, h)\ \}.
\end{array}
$$

The program (recursive decomposition)

```
                    (* R is a list of lists [A B C] or [A a] *)

let rec derive(A,f) =        (* returns true iff A -->* f *)
if (length(f)==1 and isletter(hd(f)))
  then isrule([A;(hd f)])
  else itersplit(A,[hd(f)],tl(f))

and
itersplit = function   (* search a correct splitting pour A-->f *)
 |(A,g,[])     -> false
 |(A,g,(a::h)) -> if iterrule(A,R,g,(a::h))
                    then true
```

```
                    else itersplit(A,append(g,[a]),h)

and
iterrule = function     (* search a correct rule for A-->gh *)
 |(A,[],_,_)      -> false
 |(A,r::R1,g,h)  -> if (A==hd(r) and length(tl(r))==2)
                          then if derive(hd(tl(r)),g) and
                                  derive(hd(tl(tl(r))),h)
                                 then true
                                 else iterrule(A,R1,g,h)
                          else iterrule(A,R1,g,h);;
```

Cost : Catalan numbers : $C_n = \frac{1}{2n+1}\binom{2n}{n} \approx \frac{1}{\sqrt{\pi}}4^n\, n^{-3/2}$

Exponential running time !

Improving speed : Dynamic Programming : Cocke, Kasami, Younger.

Fill in an array methodically

$$T[1,i] = \{A \mid A \to f_i \in \mathcal{R}\}.$$

$$\begin{aligned} T[2,i] &= \{A \mid A \to f_i f_{i+1} \in \mathcal{R}\} \\ &= \{A \mid \exists\ A \to BC \in \mathcal{R} \ \&\ B \in T[1,i] \ \&\ C \in T[1,i+1]\}. \end{aligned}$$

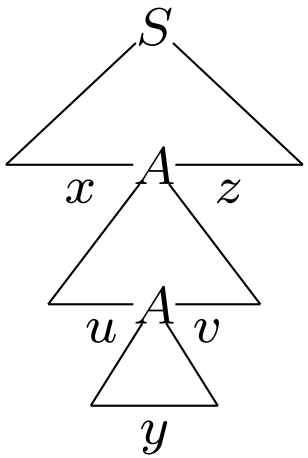
$$
\begin{aligned}
T[3,i] &= \{A \mid A \to f_i f_{i+1} f_{i+2} \in \mathcal{R}\} \\
&= \{A \mid \exists\ A \to BC \in \mathcal{R}\ \& \\
&\quad ((B \in T[1,i]\ \&\ C \in T[2,i+1]) \vee ((B \in T[2,i]\ \&\ C \in T[1,i+2]))\}.
\end{aligned}
$$

and in full generality...

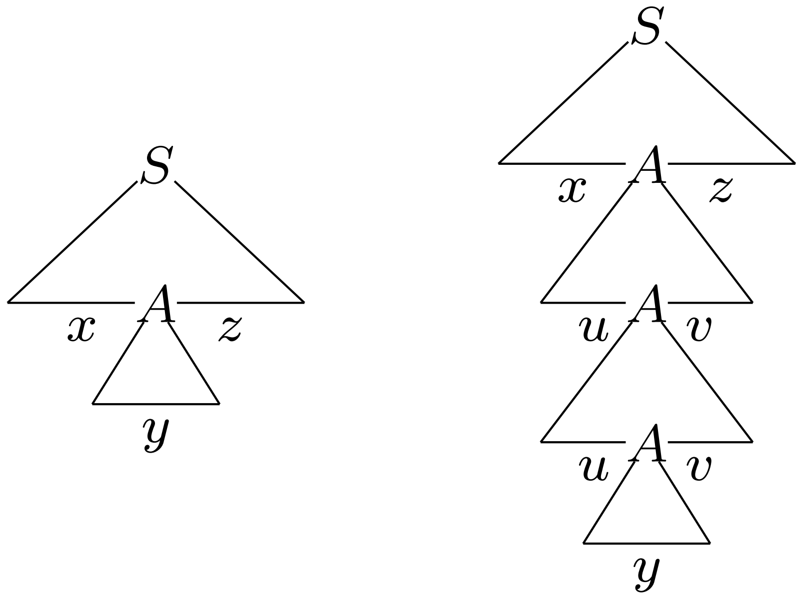
$$
\begin{aligned}
T[k,i] &= \{A \mid A \to f_i f_{i+1} ... f_{i+k-1} \in \mathcal{R}\} \\
&= \{A \mid \exists\ A \to BC \in \mathcal{R},\ \exists\ l < k,\ B \in T[l,i]\ \&\ C \in T[k-l, i+l-1]\}.
\end{aligned}
$$

Thm : *Dynamic programming achieves deterministic CF recognition in time* $O(n^3)$.

Algebraicity : the Iteration lemma for CF languages



Repetition of a variable on a branch



Thm : *[Bar-Hillel, Perles, Shamir] : Let L be a Context-Free language (infinite) ; there exists an integer λ such that for any word $w \in L$ long enough ($|w| \geq \lambda$), the word w can be decomposed as*

$$w = xuyvz, \ |uv| \geq 1,$$

and then the langage L has as a subset

$$\{xu^n yv^n z \mid n \geq 0\} \subset L.$$

Non context-free languages : $0L$-systems

Grammar $\mathcal{G} =< \Sigma, w, \mathcal{R} >$, where Σ is a finite alphabet, $w \in \Sigma^+$ is the axiom and $\mathcal{R}$ is the set of rules — finite subset of $\Sigma \times \Sigma^*$.

No distinction a priori between terminal non-terminal letters.
Non-determinism is possible

Example :

$$\mathcal{F} =< \{\diamond, \circ\}, \circ, \{\circ = \diamond, \diamond = \circ\diamond\} > .$$

Some words $\mathcal{L}(\mathcal{F})$:

$$f_0 \; : \; \circ$$

$$
\begin{array}{rcl}
f_1 & : & \diamond \\
f_2 & : & \circ\ \diamond \\
f_3 & : & \diamond\ \circ\ \diamond \\
f_4 & : & \circ\ \diamond\ \diamond\ \circ\ \diamond \\
f_5 & : & \diamond\ \circ\ \diamond\ \circ\ \diamond\ \diamond\ \circ\ \diamond \\
\text{etc.} & &
\end{array}
$$

Hence $f_{n+2} = f_n f_{n+1}$

Very restricted family !

$E0L$-systems

Grammar with terminal AND non-terminal letters $\mathcal{G} =< \Sigma \cup V, w, \mathcal{R} >$, where Σ et V are finite alphabets, $w \in (V \cup \Sigma)^+$ is the axiom and $\mathcal{R}$ is the set of rules — finite subset of $(V \cup \Sigma \times (V \cup \Sigma)^*$. Words are derived from the axion w as previously (parallel rewriting) ; the language is restricted to Σ^*.

Example :
$V = \{S, A, B, C, \overline{A}, \overline{B}, \overline{C}, N\}$ variables, $\Sigma = \{a, b, c\}$ final alphabet, S axiom and

$$\begin{aligned}\mathcal{R} = \{&S = ABC, A = A\overline{A} + a, B = B\overline{B} + b, C = C\overline{C} + c,\\ &\overline{A} = \overline{A} + a, \overline{B} = \overline{B} + b, \overline{C} = \overline{C} + c\\ &a = N, b = N, c = N, N = N\},\end{aligned}$$

$$L = \{a^n b^n c^n \mid n \geq 1\}$$

non context-free.

This language is the intersection of two CF languages.

Other languages of a similar type :

$$L = \{a^n b^p c^n d^p \mid n, p \geq 1\}$$

non context-free. A simplification of pseudoknots in RNAs...

$ET0L$-systems

Rules are organized in subsets, $\mathcal{R} = \mathcal{R}_1 \cup \mathcal{R}_2 \cup ... \cup \mathcal{R}_p$, called *tables*
Parallel rewriting, using rules of the same table !

Terminal alphabet $\Sigma = \{A, N, 1, (,)\}$, non-terminal alphabet, $V = \{S, R, T, F\}$, axiom $w = S$, (α any symbole)

$$\mathcal{R}_1 = \begin{cases} S = (xAyAz)S + (xAyAz), & \text{for } x, y, z \in \{T, NT, F, NF\} \\ & \text{and at least one of } x, y, z \in \{T, NF\}, \\ \alpha = \alpha, & \text{for } \alpha \neq S\,; \end{cases}$$

$$\mathcal{R}_2 = \begin{cases} S = R,\ F = 1F,\ T = 1 + 1T, \\ \alpha = \alpha,\ \text{for } \alpha \neq S, T, F\,; \end{cases}$$

$$\mathcal{R}_2 \;=\; \begin{cases} S = R,\ T = 1T,\ F = 1 + 1F, \\ \alpha = \alpha,\ \text{for } \alpha \neq S, T, F. \end{cases}$$

$$(11AN1AN111)(AN11AN1111A111)(1A11AN1111),$$

which can be interpreted as an encoding of boolean formula

$$(x_2 + \overline{x}_1 + \overline{x}_3)(\overline{x}_2 + \overline{x}_4 + x_3)(x_1 + x_2 + \overline{x}_4)$$

Hence we have an NP-complete language in the class! As difficult as folding for globular proteins!!!

Other mechanisms : Contextual Grammars

With the standard conventions

$$
\begin{array}{lcl@{\qquad\qquad}lcl}
S & \rightarrow & aSBC & bB & \rightarrow & bb \\
S & \rightarrow & aBC & bC & \rightarrow & bc \\
CB & \rightarrow & BC & cC & \rightarrow & cc \\
aB & \rightarrow & ab & & &
\end{array}
$$

Solution :
$L = \{a^n b^n c^n \mid n \geq 1\}$ non context-free

In the rules Righthand sides are longer than Lefthand sides

Recognition can be performed in Non-deterministic linear Space : and usually Exponential deterministic Time