Recent Trends in Formal Language Theory

Prof. Kamala Krithivasan,
Department of Computer Science and Engineering,
IIT Madras

Outline

Outline

Outline

Introduction

- Started in late 50's
- Has application in many areas of CS.
- Core course in the under graduate in CSE in many universities
- In PG courses, IT courses and Mathematics also, it is taught

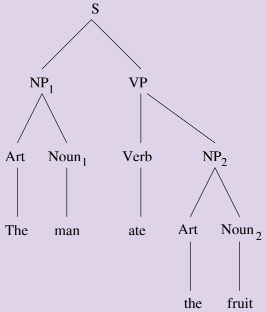
Outline

1. Introduction
2. Historical Development
   - Chomskian Hierarchy
   - Turing Machine
   - FSA
3. Recent Trends and Application
   - L System -Computer Imagery
   - Attributed Grammar
   - DNA Computing
   - Natural Computing
   - Distributed Computing
   - Contextual Grammar
   - Weighted FSA

Formal Definition of Grammar

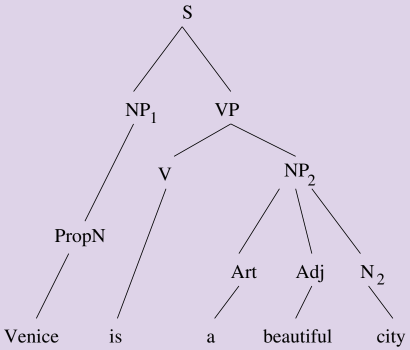
- N. Chomsky (1959) gave a definition of a grammar

Example 1: Sentence and its parse tree



Formal Definition of Grammar

Example 2: Sentence and its parse tree



Rules for “The man ate the fruit”

$$
\begin{array}{rcl}
< S > & \rightarrow & < NP_1 >< VP > \\
< NP_1 > & \rightarrow & < Art >< Noun_1 > \\
< Art > & \rightarrow & the \\
< Noun_1 > & \rightarrow & man \\
< VP > & \rightarrow & < Verb >< NP_2 > \\
< Verb > & \rightarrow & ate \\
< NP_2 > & \rightarrow & < Art >< Noun_2 > \\
< Noun_2 > & \rightarrow & fruit
\end{array}
$$

Derivation of “The man ate the fruit”

$$
\begin{aligned}
< S > \quad & \Rightarrow \quad < NP_1 >< VP > \\
& \Rightarrow \quad < Art >< Noun_1 >< VP > \\
& \Rightarrow \quad The < Noun_1 >< VP > \\
& \Rightarrow \quad The\ man < VP > \\
& \Rightarrow \quad The\ man < Verb >< NP_2 > \\
& \Rightarrow \quad The\ man\ ate < NP_2 > \\
& \Rightarrow \quad The\ man\ ate < Art >< Noun_2 > \\
& \Rightarrow \quad The\ man\ ate\ the < Noun_2 > \\
& \Rightarrow \quad The\ man\ ate\ the\ fruit
\end{aligned}
$$

Rules for "Venice is a beautiful city"

$$
\begin{array}{rcl}
< S > & \rightarrow & < NP_1 >< VP > \\
< NP_1 > & \rightarrow & < PropN > \\
< PropN > & \rightarrow & Venice \\
< VP > & \rightarrow & < Verb >< NP_2 > \\
< Verb > & \rightarrow & is \\
< NP_2 > & \rightarrow & < Art >< adj >< N_2 > \\
< Art > & \rightarrow & a \\
< adj > & \rightarrow & beautiful \\
< N_2 > & \rightarrow & City
\end{array}
$$

Context free grammar

A context free grammar is a 4-tuple $G = (N, T, P, S)$, where N is a finite set of nonterminal symbols called the nonterminal alphabet, T is a finite set of terminal symbols called the terminal alphabet, $S \in N$ is the start symbol and P is a set of productions (also called production rules or simply rules) of the form $A \rightarrow \alpha$, where $A \in N$ and $\alpha \in (N \cup T)^*$

Context free grammar

Derivations

If $\alpha A \beta$ is a string in $(N \cup T)^*$ and $A \rightarrow \gamma$ is a rule in P, from $\alpha A \beta$ we get $\alpha \gamma \beta$ by replacing A by γ. This is denoted as $\alpha A \beta \Rightarrow \alpha \gamma \beta$. $\Rightarrow$ is read as 'directly derives'. If $\alpha_1 \Rightarrow \alpha_2$, $\alpha_2 \Rightarrow \alpha_3, \ldots, \alpha_{n-1} \Rightarrow \alpha_n$, the derivation is denoted as $\alpha_1 \Rightarrow \alpha_2 \Rightarrow \cdots \Rightarrow \alpha_n$ or $\alpha_1 \stackrel{*}{\Rightarrow} \alpha_n$. $\stackrel{*}{\Rightarrow}$ is the reflexive, transitive closure of $\Rightarrow$.

Context free language

The language generated by a grammar $G = (N, T, P, S)$ is the set of terminal strings derivable in the grammar from the start symbol.

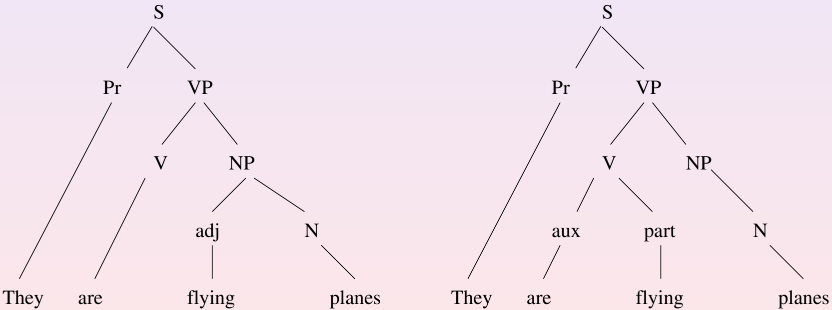
$$L(G) = \{w/w \in T^*, S \overset{*}{\Rightarrow} w\}$$

$G = (N, T, P, S)$ where $N = \{S, A\}$, $T = \{a, b, c\}$, production rules in P are

1. $S \rightarrow aSc$
2. $S \rightarrow aAc$
3. $A \rightarrow b$

The Language generated by G: $L = \{a^n bc^n | n \geq 1\}$

Ambiguity in context free grammar



Outline

Turing Machine

- Defined by A. M. Turing in 1936
- Abstract model of computation
- Still considered as the model of computation
- Has stood the test of time

Undecidability

- For certain problems, no algorithm exists
  - Shown by A. M. Turing
  - A breakthrough concept

Halting problem -undecidable

- Consider the set of input free programs P
- Can you write a program which will take a program p in P as input and tell when p will halt or loop (no limit on time or memory)
- Turing showed that there can not exist a program which will do this

Proof of undecidable of Halting problem

Suppose such a program Halt exists
Halt(p)='yes' if p halts
Halt(p)='no' if p loops
ABSURD:

```
begin
    If Halt(ABSURD) = 'yes'
        then begin
            while 'true'
            print 'ha'
            end;
        else stop;
end
```

Proof of undecidable of Halting problem

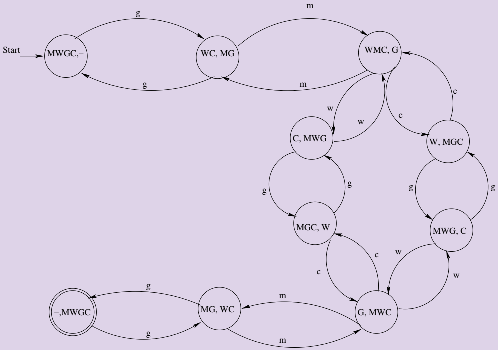
Halt(ABSURD)	
halts	does not halt
does not halt	halts

Outline

1. Introduction
2. Historical Development
   - Chomskian Hierarchy
   - Turing Machine
   - FSA
3. Recent Trends and Application
   - L System -Computer Imagery
   - Attributed Grammar
   - DNA Computing
   - Natural Computing
   - Distributed Computing
   - Contextual Grammar
   - Weighted FSA

FSA

FSA for man, wolf, goat and cabbage problem



FSA -example

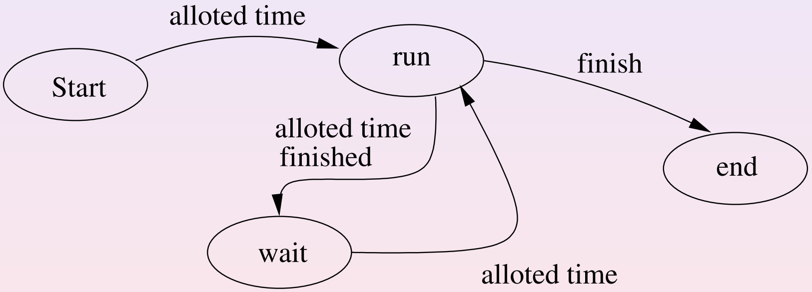
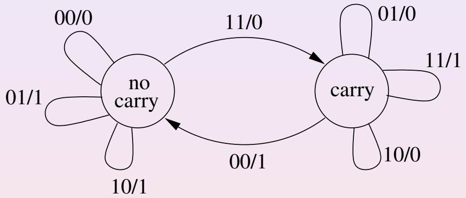


Figure: Transition diagram of process

FSA -example



	1	0	0	1	1
	1	0	0	1	1
1	0	0	1	1	0

Formal definition of FSA

- A NDFSA is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$
- Q is set of states, Σ is set of inputs
- δ is a map from $Q \times \Sigma$ to 2^Q
- F is subset of Q, called as the set of all final states

DFSA

- if the range of δ in the above defn is replaced by Q, the FSA is called as DFSA

AFL Theory

Six bsic operations

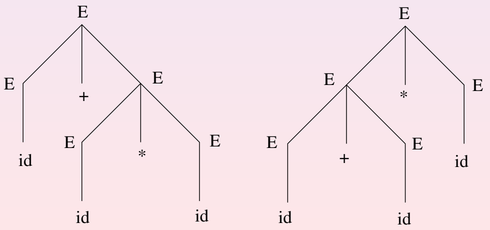
- Union
- Concatenation
- ϵ-free kleene closure
- ϵ-free homomorphism
- Intersection with regular sets
- Inverse homomorphism

Useful in compiler design

- Compilar is a program which translates a high level program into machine code
- It has two parts namely analysis and synthesis
- Lexical Analyzer -FSA
- Parser -PDA

Parsing arithmetic expression

In arithmetic expression, it is better to avoid ambiguity
$E \rightarrow E + E$, $E \rightarrow E * E$
$E \rightarrow id$
will generate $id + id * id$ which will have two different derivation trees

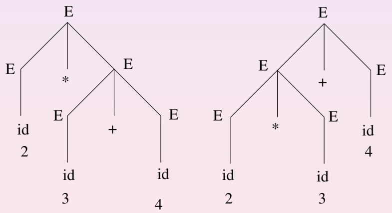


Parsing arithmetic expression

In arithmetic expression

$E \rightarrow E + E, E \rightarrow E * E$

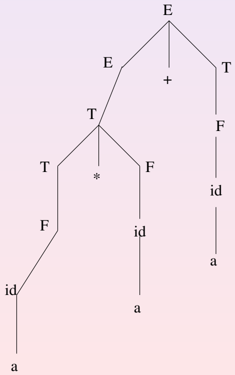
$E \rightarrow (E), E \rightarrow id$



Left tree: $2 * (3 + 4) = 14$ Right tree: $(2 * 3) + 4 = 10$

Parsing arithmetic expression

$$E \rightarrow E + T,\ E \rightarrow T,\ T \rightarrow T * F,\ T \rightarrow F,\ F \rightarrow (E),\ F \rightarrow id$$



L-Systems- Biological Motivation

L systems were defined by A. Lindenmayer in an attempt to describe the development of multicellular organisms. In the study of developmental biology, the important changes that take place in cells and tissues during development are considered. *L* systems provide a framework within which these aspects of development can be expressed in a formal manner. *L* systems also provide a way to generate interesting classes of pictures by generating strings and interpreting the symbols of the string as the moves of a cursor.

L-Systems- Biological Motivation

From the formal language theory point of view, L systems differ from the Chomsky grammars in three ways.

- Parallel rewriting of symbols is done at every step. This is the major difference.
- There is no distinction between nonterminals and terminals (In extended L system we try to introduce the distinction).
- Starting point is a string called the axiom.

L-Systems - Example

Consider the following DP0L system

$$\pi_2 = (\Sigma, 4, P)$$

where $\Sigma = \{0, 1, 2, \ldots, 9, (,)\}$
P has rules
$0 \rightarrow 10$, $1 \rightarrow 32$, $2 \rightarrow 3(4)$,
$3 \rightarrow 3$, $4 \rightarrow 56$, $5 \rightarrow 37$,
$6 \rightarrow 58$, $7 \rightarrow 3(9)$, $8 \rightarrow 50$,
$9 \rightarrow 39$ $(\rightarrow ($, $) \rightarrow)$

L-Systems - Example

Ten steps in the derivation are given below

1 4, 2 56, 3 3758

4 33(9)3750, 5 33(39)33(9)3710

6 33(339)33(39)33(9)3210

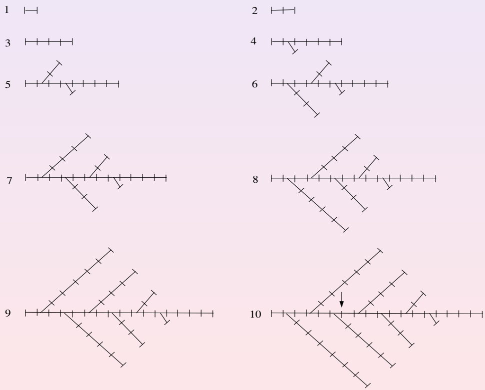
7 33(3339)33(339)33(39)33(4)3210

8 33(33339)33(3339)33(339)33(56)33(4)3210

9 33(333339)33(33339)33(3339)33(3758)33(56)33(4)3210

10 33(3333339)33(333339)33(33339)33(33(9)
3750)33(3758)33(56)33(4)3210

L-Systems - Example



Array Grammars -Digital Pictures

- Like strings, rectangular arrays of symbols are generated
- Useful for describing pictures

Regulated Rewriting

Putting control on the manner of applying the rules to increases the generative capacity in some cases

- Matrix grammar
- Time varying grammars
- Programmed grammars
- Control sets
- Random context free grammars
- Ordered grammars
- Indian parallel grammars

Graph Grammars

- Generates grammar
- useful in pattern recognition, incremental compilers, etc.

Cellular Automata

- Defined by Van Neuman
- Parallel device
- Useful in many fields

Outline

L System - Computer Imagery

Many fractals can be thought of as a sequence of primitive elements. These primitive elements are line segments. Fractals can be coded into strings. Strings that contain necessary information about a geometric figure can be generated by *L*-systems. The graphical interpretation of this string can be described based on the motion of a LOGO-like turtle.

L System - Computer Imagery

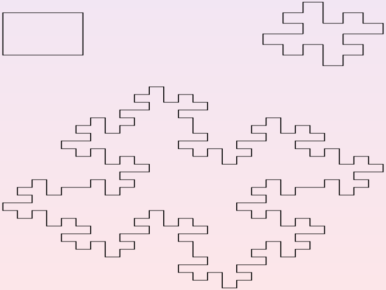
Interpretation of a string

Let S be a string and (x_0, y_0, A_0) be the initial state of the turtle, and step size d, angle increment δ are the fixed parameters. The pattern drawn by the turtle corresponding to the string S is called the turtle interpretation of the string S.

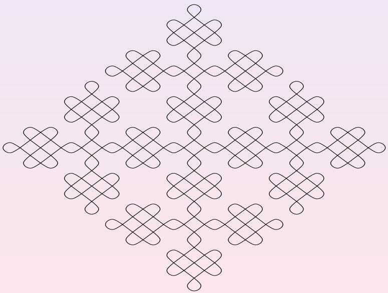
L System - Computer Imagery

Axiom : $w : f + f + f + f$

production : $f \to f + f - f - ff + f + f - f$



L System - Computer Imagery

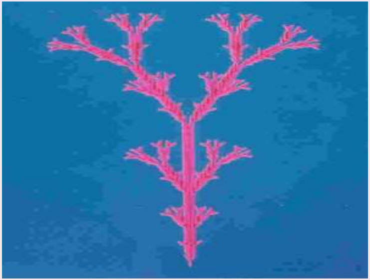


L System - Computer Imagery

Generation of Plant Structure

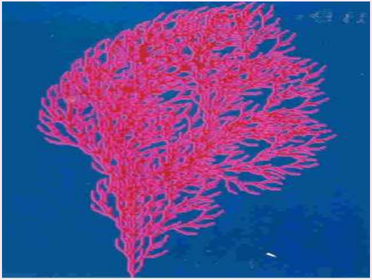
L System - Computer Imagery

Generation of Plant Structure



L System - Computer Imagery

Generation of Plant Structure



L System - Computer Imagery

Generation of Plant Structure

Outline

1. Introduction
2. Historical Development
   - Chomskian Hierarchy
   - Turing Machine
   - FSA
3. Recent Trends and Application
   - L System -Computer Imagery
   - **Attributed Grammar**
   - DNA Computing
   - Natural Computing
   - Distributed Computing
   - Contextual Grammar
   - Weighted FSA

Attributed Grammar

- Code Generation
- Network Load Modeling

Outline

Abstract models to represent the recombinant behavior of DNA strands

- Splicing System
- Sticker System
- Watson-Krick Automata

Outline

Natural Computing

- Membrane Computing
- Peptide Computing
- Immune Computing

Outline

1. Introduction
2. Historical Development
   - Chomskian Hierarchy
   - Turing Machine
   - FSA
3. Recent Trends and Application
   - L System -Computer Imagery
   - Attributed Grammar
   - DNA Computing
   - Natural Computing
   - Distributed Computing
   - Contextual Grammar
   - Weighted FSA

Distributed Computing

- Grammar System
  - CD Grammar systems (Black Board Model)
  - PC Grammar systems (Class Room Model)

Outline

An application of contextual grammar

- Natural Language Processing

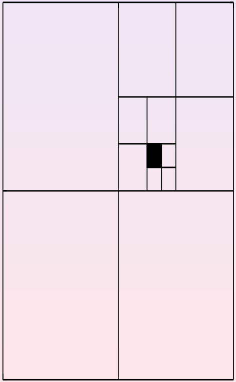
Outline

1. Introduction
2. Historical Development
   - Chomskian Hierarchy
   - Turing Machine
   - FSA
3. Recent Trends and Application
   - L System -Computer Imagery
   - Attributed Grammar
   - DNA Computing
   - Natural Computing
   - Distributed Computing
   - Contextual Grammar
   - Weighted FSA

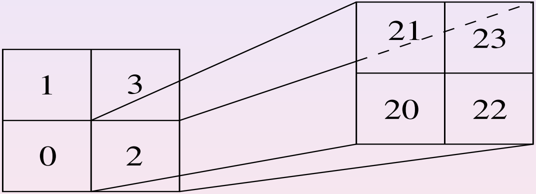
Weighted FSA

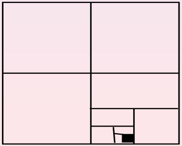
1	3
0	2

11	13	31	33
10	12	30	32
01	03	21	23
00	02	20	22



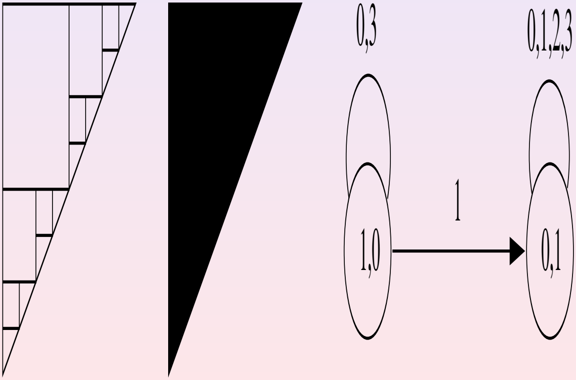
Weighted FSA



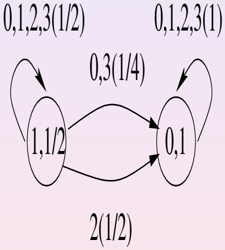


Address of Black Square is 2022

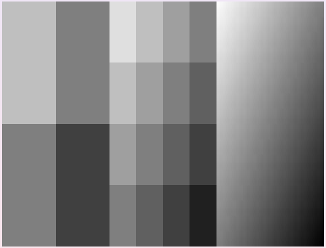
Weighted FSA



Weighted FSA

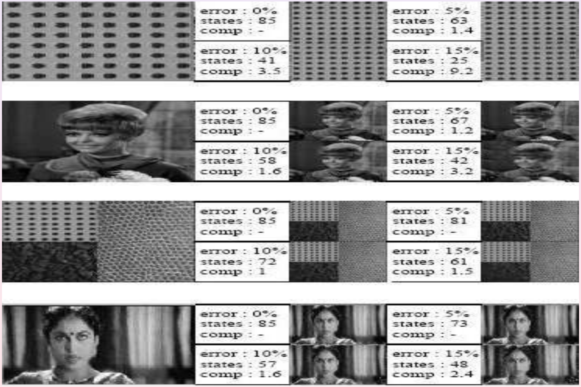


WFA



2x2 4x4 128x128

Weighted FSA



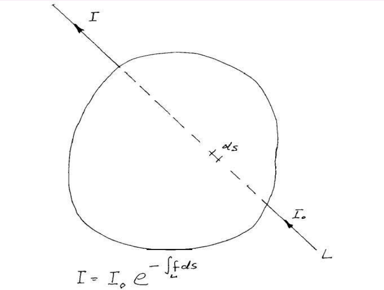
Outline

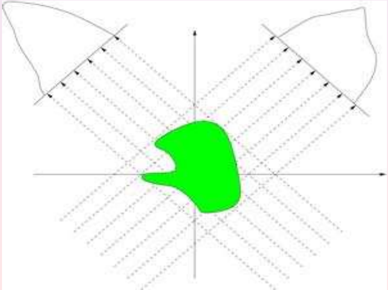
1. Introduction
2. Historical Development
   - Chomskian Hierarchy
   - Turing Machine
   - FSA
3. Recent Trends and Application
   - L System -Computer Imagery
   - Attributed Grammar
   - DNA Computing
   - Natural Computing
   - Distributed Computing
   - Contextual Grammar
   - Weighted FSA

What is Tomography

Tomography

- Study of reconstruction of 2D-slice of 3D-objects from its projections





What is Discrete Tomography

Binary image

$$A = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix} B = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

Two Projections

- Projection along row: $R = (3,\ 3,\ 4,\ 3)$
- projection along column: $C = (3,\ 4,\ 4,\ 2)$

Thank You