**Syntactic Pattern Recognition**

1. Importance of Syntactic Pattern Recognition

Statistical pattern recognition attempts to classify patterns based on a set of extracted features and an underlying statistical model for the generation of these patterns. Ideally, this is achieved with a rather straight-forward method:  
1. determine the feature vector,  
2. train the system,   
3. classify the patterns  
However, patterns that include structural or relational information are difficult to represent as feature vectors. Syntactic pattern recognition uses this structural information for classification and description.

A pattern can be described by a hierarchical structure of sub patterns, analogous to the syntactic structure of formal languages. This permits the use of formal languages for tackling pattern recognition problems. A pattern grammar consists of finite sets of elements called variables, primitives, and productions. The production rules determine the type of grammar to be used for pattern recognition. Among the most studied grammars are regular grammars, context-free grammars, and context-sensitive grammars. The selection of pattern primitives, the assembling of the primitives and their relationships into pattern grammars, and analysis and recognition uses the rules of these grammars. This approach is useful in dealing with patterns, which cannot be conveniently described by numerical measurements.

Among the various techniques for object recognition, syntactic pattern recognition technique is generally preferred when high-speed recognition is a matter of concern. The idea behind syntactic pattern recognition is the specification of a set of pattern primitives, a set of rules that governs their interconnection and a recognizer whose structure is determined by the set of rules in the grammar. The description of an object is called pattern. A description of the pattern structure is useful for recognizing entities when a simple classification isn’t possible. It can also describe aspects that cause a pattern not to be assigned to a particular class. In complex cases, recognition can only be achieved through a description for each pattern rather than through classification.

2. Use of Formal Language Theory in Syntactic Pattern Recognition  
  
Syntactic pattern recognition follows the theory of formal languages. The origin of formal language theory may be traced in middle 1950’s with the development by Noam Chomsky’s mathematical model of a grammar related to his work in natural languages. The concepts which are helpful to comprehend the formal language theory are defined below:   
• An Alphabet is any finite set of symbols.   
• A word over an alphabet is any string of finite length composed of symbols from the alphabet. For example, valid words of alphabet {0,1} are 0,1,00,01,10,11….   
• A word with no symbols is called empty word and denoted by λ.  
• A language is any set of words over an alphabet.   
• As every language follows some specific grammar. Similarly the formal language is associated with a grammar which is basically a 4-tuple: G = {VN, VT, P, S}   
Where, VN is a set of non-terminals (variables);   
 VT is a set of terminals (constants);   
 P is a set of productions or rewriting rules;   
 S is a set start or root symbol.   
S belongs to the set VN and VN and VT are disjoint sets, whereas V is the union of sets VN and VT. V\* denotes the set (free monoid) of words consisting of the empty word L whereas V+ is a set (free semi group) of sentences of V\*-λ.

The language generated by G, is denoted by L(G). It is the set of strings that satisfy two conditions:   
(i) Each string is composed only of terminals (i.e., each string is a terminal sentence).  
(ii) Each string can be derived from S by suitable application of production from the set P.

The set P of production consists of expressions of the form α→β. The symbol → indicates replacement of string α by the string β, where α is a string in V+ and β is a string in V\*. The set of production formulas are a part of a normal algorithm whose concept was introduced by A.A. Markov. The normal algorithm recognizes angles from changes in the direction during contour tracking. This is done by Look Ahead Tracing (LAT) technique.

Grammars differ only in their productions. Now, various types of grammar are:   
• Unrestricted Grammar: It has production formulas of the form α→β, where α is a string and β is another string.   
• Context-Sensitive Grammar: It has production of the form α1Aα2→α1βα2, where α1 and α2 are in V\*, β is in V+ and A is in VN. This grammar allows replacement of the non-terminal A by the string β only when A appears in the context α1Aα2 of string α1 and α2.  
• Context-free Grammar: It has production of the form A→β, where A is in VN and β is in V+. The name context free arises from the fact that the variable A may be replaced by a string β regardless of the context in which A appears.   
• Regular (or Finite-state) Grammar: It is one with productions of the form A→αB or A→α, where A and B are variables in VN and a is a terminal in VT .

These grammars are sometimes called type 0, 1, 2 and 3 grammars respectively.

The basic concept of underlying syntactic pattern recognition is illustrated by the development of mathematical models of computing machines, called automata. Given an input string, an automaton is capable of recognizing whether the pattern belongs to the language with which the automaton is associated.

A finite automaton is defined as the 5-tuple, Af = (Q, ∑, δ, q0, F)   
Where Q is a finite, nonempty set of states,   
∑ is a finite input alphabet,   
δ is a mapping from Q X ∑ into the collection of all subsets of Q,   
q0 is the starting state, and  
F is a set of final or accepting states.

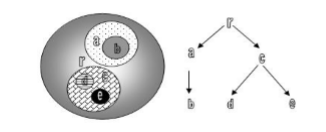
For syntactic pattern recognition, the simplest sub-patterns are called pattern primitives, and are much easier to recognize than the overall patterns. The language used to describe the structure of the patterns in terms of sets of pattern primitives is called the pattern description language. The pattern description language will have a grammar that specifies how primitives can be composed into patterns. When a primitive within the pattern is identified, syntax analysis (parsing) is performed on the sentence describing the pattern to determine if it is correct with respect to the grammar. Syntax analysis also gives a structural description of the sentence associated with the pattern. One advantage of this approach is that a grammar (rewriting) rule can be applied many times.

Example: A grammar describing four blocks arranged in 2-block stacks:  
VT = { table , block , + , ↑ } (terminal symbols)  
VN = { DESK,LEFT STACK,RIGHT STACK} (non-terminal symbols)   
S = DESK ∈ VN  (root symbol)  
P = {DESK → LEFT STACK + RIGHT STACK  
 DESK → RIGHT STACK + LEFT STACK   
 LEFT STACK → block ↑ block ↑ table  
 RIGHT STACK → block ↑ block ↑ table} (production rules)

3. Syntactic Pattern Description

Let the object to be recognized in an image is a two dimensional pattern. The string grammars of this pattern can be obtained by simple juxtaposition of a string, to form new strings. Juxtaposition of two strings means placing the objects together, without losing the identity of the objects. Concatenation can also be done but it involves spatial rearrangement as well as a loss of identity on the part of the individual objects. Juxtaposition of structures takes place only at two points called a head and tail of an arrow defined by these two points. Graph-like patterns can be recognised as two-dimensional patterns which can then be reduced to an equivalent string representation.

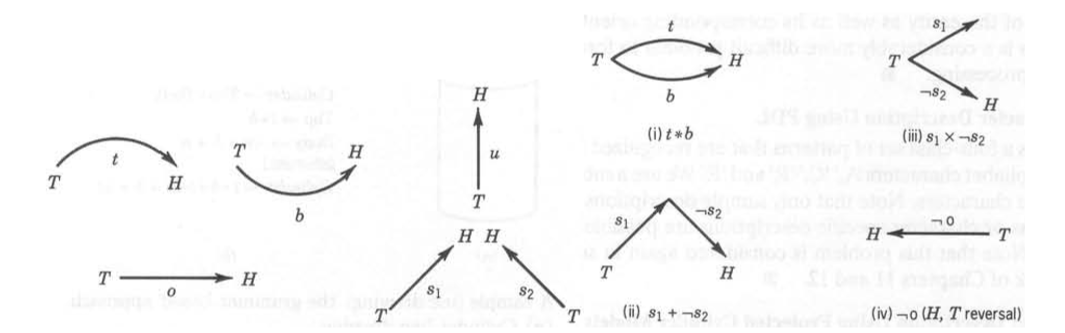
The other useful technique for describing two-dimensional relationship is based on tree structures. A tree is a finite set T of one or more nodes such that:   
• There is an especially designated node called the root of the tree.  
• The remaining nodes (excluding the root) are partitioned into m disjoint sets T1, T2……, Tm, m‡0, where each of these sets is in turn a tree. These trees are called sub trees of the root. A node of degree zero is called a leaf, while a node of higher degree is called a branch node. The tree representation of a pattern is called a pattern tree.



Tree representation of patterns

Syntax-directed grammar is a mechanics for determining whether or not a pattern can be generated by a particular grammar. Once the grammars are known the basic problem is the development of a procedure for determining whether or not a given pattern represents a valid formula, a word or a sentence. The procedure used in formal language theory to accomplish this is called parsing. Basically two types of parsing techniques are considered: (i) Top Down and (ii) Bottom Up.

In the Top Down approach, the top or root of the (inverted) tree is the start symbol S and through repeated application of the productions of the grammar one can attempt to arrive at the given terminal sentence. The bottom up approach on the other hand starts with the given sentence and attempts to arrive at the symbol S by applying the production. In either case if the parsing fails then the given pattern represents an incorrect sentence and is therefore rejected. The parsing process can be further improved by employing the rules of syntax of the grammar. Syntax is defined as the juxtaposition and concatenation of object. A rule of syntax states some permissible (or prohibited) relation between objects. A syntax-directed parser employs the syntax of the grammar in the parsing process.

Example: Representation of a Cylinder using Line Drawing Picture Description Grammar  


A set of terminal symbols {t, b, u, o, s, \*, -, +} where   
+ represents head to tail concatenation,   
\* represents head-head and tail-tail attachment,   
– represents the head-tail reversal. (x should be \* in (iii) )

