SOME EXAMPLES OF TABULAR ALGORITHMS IN SYNTACTIC PATTERN RECOGNITION

T.Bellone*,E. Borgogno *, G. Comoglio *
*DIGET, Politecnico di Torino, Torino, Italy
tamara.bellone@polito.it, enrico.borgogno@polito.it, giuliano.comoglio@polito.it

Commission VI, Working Group 3

KEY WORDS: Pattern Recognition, Image understanding, Algorithms

ABSTRACT:

Image understanding is a very important step in the course of data analysis in GIS. Syntactic Pattern Recognition of models is a process, as used in Cartography and Remote Sensing, which makes possible the matching of parts of maps, of images and 3D models with archetypes and patterns (parsers). The main item of the entire process is the so-called Parsing, a device used in Linguistics, successively used in Cognition Disciplines. Simply, it must help decide whether a data string (a phrase) can be a part of an existing pattern (language).
A number of algorithms are available for Parsing, for the needs of specific grammars: although not suited for any grammars, tabular methods help save time, as the Kasami method, remarkably simple to use: it works well in the case of context-free grammars, as reduced to the so- called Chomsky's normal form.

1. Foreword

The so-called computer science revolution has also taken place in the area of geodetic sciences, which eventually are playing a key role in it (Geomatics).
Even common people know that many mental preocesses are based upon evaluation of contrast and difference: this is, for instance, the art of seeing, and generally a good approach in all cognitive sciences, also in Linguistics. Language, indeed, is based upon blending of discrete parts (phonemes, morphemes). Also, vision and speaking are based upon principles not quite different. This is why some present procedures of Geomatics may be referred to logical and symbolic structures proper for Mathematical Logic and Linguistics. Some improvements in GIS and Digital Photogrammetry are referred to as Computer Vision, Image Processing, Image Understanding, Machine Learning, which are linked to developments of Artificial Intelligence.
An easy case of this cultural melting is Syntactic Pattern Recognition: it is a procedure, widely used in Cartography and Remote Sensing, that trusts upon matching of sections of maps and/or images or 3D models with archetypes or objects (parsers). Also, parsing is a topic proper of Linguistics, which has been borrowed from cognitive sciences; Artificial Intelligence in turn is based upon Logic and Linguistics (Mathematical Logic and Mathematical Linguistics).
A Syntactic Pattern Recognition system consists with three parts: pre-processing, pattern description and syntax analysis.
Pre-processing includes pattern encoding and approximation, filtering, restoration and enhancement. The pattern representation procedure consists of pattern segmentation and feature (primitive) extraction, in order to represent a pattern in terms of its sub-patterns and pattern primitives: each sub-pattern is identified by a given set of pattern primitives.
The decision whether or not the representation belongs to the class of patterns described by the given grammar or syntax (is syntactically correct) is made by a "parser".
Parsing is then the syntax analysis: this is an analogy between the hierarchical (treelike) structure of patterns and the syntax of language. Patterns are built up by sub-patterns in various ways of composition, just sentences are built up by words and sub-patterns are built up by concatenating primitives (features) just words by concatenating characters (Sester, 1990).
Parsing adaption to the GIS context can be thought for selecting reference shapes from numerical cartography or raster geocoded images. Phrases are cartographic objects and language is the set of rules which define these objects satisfy or not selecting criteria.
A double use of Parsing is possible in the framework of GIS: both at the generation stage, and for the process of query and data analysis.
The shape reconnaisance by parsing is of evident interest for:

- map production, implementation and updating through automatic extraction of shapes from raster data;
- editing of existing maps (as association of codes to not-still-coded shapes or perceiving of difference between a closed polygon and an open line in order to close it);
- query purposes, aimed to the extraction of raster features or not-coded shapes from the database; an ample variety of questions may be posed, as one deals with new geometrical features which can be classified as recursive shapes according to a reference pattern.

In this paper a preliminary investigation of this problem is done, showing some different approaches, and a brief example of recognition of simple geometrical shape is given.

2. The Theory of Formal Languages

In the theory of formal languages, a language is defined as a set of strings: a string is a finite sequence of symbols chosen from some finite vocabulary. In natural languages, a string is a sentence, and the sentences are sequences of words chosen from some vocabulary of possible words. A grammar is defined as a four-tuple:

$$G = (N, T, P, S)$$

where N and T are the non terminal and terminal vocabularies of G, P is the set of production rules, and S is the start symbol.

A formal language is indeed defined by:

- A terminal vocabulary of symbols (the words of the natural language)
- A non terminal vocabulary of symbols (the syntactic categories, e.g. noun, verb)
- A set of productions (the phrase structure rules of the language)
- The so called start symbol

We start from a special non terminal S, and S is replaced by the string on the right side of a chosen production rule. The process of rewriting a substring according to one of the rewriting production rules continues until the string consists only of terminal symbols:

$S \rightarrow aS|\ bS|\ \varepsilon$

where the symbol | indicates "or" and ε is the null string. The succession of strings that result from the process is a derivation from the grammar: to find a derivation (a parse) for a given sentence (sequence) is called parsing.

As to the latter approach, let's remind that in the Theory of Formal languages Chomsky divides languages in classes, thus forming a hierarchy of languages, based upon different grammars:

- Unrestricted grammars (0-type grammars)
- Context-sensitive grammars (1-type grammars)
- Context-free grammars (2-type grammars)
- Regular grammars or finite state grammars (3-type grammars)

The most general grammar is obviously the 0-type, which bears no limits for rewriting rules: for the other types, such restrictions are regularly increasing. Types 0 and 1 are able to describe natural languages as the other two, much simpler to manage from a computational viewpoint, are more suited into limited backgrounds and have been hitherto used for artificial languages.

In the 1-type grammar, rewriting rules restraints bear that the right side of a rule should have at least as many symbols as the left side;

For the 2-type grammar, all rules should have just one non-terminal symbol on the left side

For 3-type grammar, the right side has only one terminal symbol, or one terminal symbol and a non-terminal one for every production rule.

The language classes as arranged by Chomsky need a number of reconnaissance devices (automata):

0-type languages: Turing machines

1-type languages: bounded automata

2-type languages: pushdown automata

3-type languages: finite state automata.

Although many classes of patterns appear context sensitive, context sensitive grammars have rarely been used, because of their complexity. Context free programmed grammars have been used , probably due to their effectiveness in describing natural languages: they are able to capture much of natural and artificial languages, but many problems required extensions.

Context free grammars cannot model all the characteristics of natural languages. One example is the conversion of sentences from active to passive voice, Chomsky (1957) developed the theory of transformational grammar, in which a sentence is derived as a deep structure, then modified by transformational rules and finally converted in surface form by phonological rules. The deep structure is derived by a context free grammar, which generates "proto" phrases: strings like "the bridge crosses the river" and "the river is crossed by the bridge" have the same deep structure, but a different surficial structure.

In syntactic pattern recognition problems, it is often important to represent the two or three dimensional structure of sentences in the languages. Traditional context free grammars generate only one dimensional string. Context free graph grammars have been developed in order to construct a graph of terminal nodes instead of a string of terminal symbols.

A statistic approach to language started as the scholars realized the value of statistic methods in the recognition of speech. Hidden Markov's model was first used by the same for modelling the language of "Evgenij Onegin" (Markov, 1913). A grammar normally divides strings in just two classes: the grammatically correct ones, and the others. In any case, ambiguity is very frequent, and it is even deliberately pursued as sometimes happens in poetry. The simplest way of accounting for ambiguity is the usage of a stochastic grammar.

3. Parsing Algorithms

In accordance with the needs of different grammars, a certain number of Parsing algorithms are in use.

Parsing procedures may be top-down or bottom-up type, as one starts from initial symbol S, operating substitutions until only terminal symbols are present, such as to fit the clause, or as one starts from the string backward till the start symbol S.

Tabular procedures are time saving, however they are not suitable to all grammar types.

As an example, in the following the tabular Kasami method is used, which is remarkable for its semplicity: it works with context-free grammars, previously reduced to the Normal Chomsky Form.

Actually, some procedures use simplified context-free grammars, based upon certain theorems, one of which being the so called Chomsky's Normal Form Theorem (CNF): it states that a context free language may be generated from a grammar, so that the production rules shall have one of the following forms:
A→BC, B and C being non-terminal symbols
A →*a*, *a* being a terminal symbol.
For instance, let us make up the CNF, starting with the context-free grammar:

$S \rightarrow bA$ $S \rightarrow aB$ $A \rightarrow a$
$B \rightarrow b$ $A \rightarrow aS$ $B \rightarrow bS$
$A \rightarrow bAA$ $B \rightarrow aBB$

Some well known algorithms may be used to convert a general context free grammar into a Normal Form according to Chomsky.
Rules $A \rightarrow a$ and $B \rightarrow b$ should first be available in the canonical form. No rule exist in the form $C \rightarrow D$, so (non terminal) variables can be directly substituted to terminal symbols, the first rewriting rule being thus substituted by both rules:

$S \rightarrow C_1A$ $C_1 \rightarrow b$

Therefore, the following substitutions take place:

$A \rightarrow aS$ with $A \rightarrow C_2S$ $C_2 \rightarrow a$,
$S \rightarrow aB$ with $S \rightarrow C_4B$ $C_4 \rightarrow a$
$A \rightarrow bAAA$ with $A \rightarrow C_3AA$ $C_3 \rightarrow b$
$B \rightarrow bS$ with $B \rightarrow C_5S$ $C5 \rightarrow b$
$B \rightarrow aBB$ with $B \rightarrow C6BB$ $C6 \rightarrow a$

At last:
$A \rightarrow C_3AA$, with $A \rightarrow C_3D_1$ and
$B \rightarrow C_6BB$, with $B \rightarrow C_6D_2$ and $D_2 \rightarrow BB$

Once a context free garmmar has been transformed into its Chomsky's normal forma, Kasami's table can be assembled, so that one can decide whether a string belongs to the said grammar.
Be $w= a_1a_2....a_n$ a string whose pertinence to a given gramamr is to be tested, the grammar being already reduced to the CNF.
The algorithm is basically a triangular parsing table, whose elements are t_{ij} for $1 \le i \le n$ e $1 \le j \le n-i+1$. Every t_{ij} should have a value being a sub-set of N. The non terminal symbol shall be into t_{ij} if, and only if:
$A \rightarrow a_1a_{i-1}...a_{i-j-1}$. The string shall belong to the said language just in case S shall be found into t_{1n}.
The table is assembled as follows:

- one states t_{i1}=A if $A \rightarrow a_i$
- one states t_{ij} =A even for a single k, such that $1 \le k < j$, if $A \rightarrow BC$ is to be found in P, having B present in t_{ik} and C in $t_{i-k,j-k}$.

Also, let us have the grammar:

$S \rightarrow AB, S \rightarrow BC$
$A \rightarrow BA, A \rightarrow a$
$B \rightarrow CC, B \rightarrow b$
$C \rightarrow AB, C \rightarrow a$

and the baaba string.
The table which allows state whether the string belongs to the language generated by the said grammar is as follows:

i →
j ↓

B	A,C	A,C	B	A,C
S,A	B	S,C	S,A	
0	B	B		
0	S,A,C			
S,A,C				

Since S belongs to the case t_{15}, we may state that the string baaba is generated by the said grammar.

4. Application test

In order to evaluate how well such approach could be applied to the automatic interpretation of images (possibly for cartographic upgrading purposes or GIS geometric query) preliminary tests have been carried out onto simple geometric images.
It is here shown an example addressed to draft a possible operational path for Parsing based pattern recognition of simple geometric entities.
An appropriate test image has been created showing three different geometric figures: a rectangle, a scalene triangle and a equilateral triangle. The goal is to verify if the implemented grammar could correctly decide if one figure is or not an equilateral triangle.

The recognition process goes on in the following way:

- a preliminary identification, based on radiometric/spectral discriminants, of the pixels of the image probably belonging to the searched objects is firstly carried out;
- the selected pixels are then grouped in different distinct geometric entities using neighbourhood and region growing criteria (different colors in the image below);

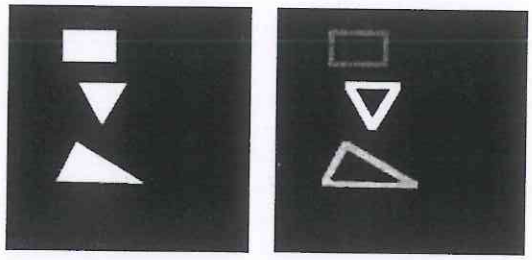


Fig. 1 Test image and polygon grouped pixels image.

- for each entity a frame window is clipped from the original image and a Förstner filtering and thresholding algorithm is applied in order to select pixels most probably representing the vertices of the figure which has to be recognized;

➢ assuming that figures are closed polygons vertices coordinates are used to define length and direction of the connecting lines. These are the geometric primitives used in the parsing grammar. A simple translation from numbers to letters (defined when grammar has been defined) allow to transfer information to the parsing engine algorithm which has to decide if the object belongs or not to the defined grammar, that is if that polygon is or not an equilateral triangle.

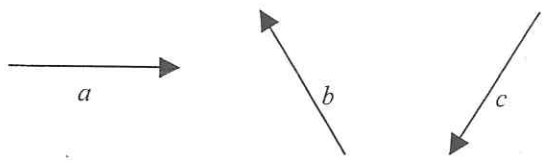
All these steps have been implemented using the IDL programming language. We do not intended to deeply describe well known image processing algorithms. Otherwise, we care to briefly describe how the parsing algorithm works. It has firstly to be structured, defining the deciding grammar. This is a static part of the program; in fact ,once defined, it never changes during the recognizing process. Changing grammar means to change the program text. In the future we intend to define a standard text file the user can fill off-line to define the different grammars he wants to use. Such file could be directly be read by the program while executing. Grammar has been structured as a matrix, with a column number equal to the number of the generic values A_i, and with a row number equal to the maximum number of values (terminal and non) that each generic value A_i can assume. Each generic value A_i (first line) determines its own column with the possible values it can assume. Terminal values are listed in the last line. The matrix is a sparse one.

The program reads the string corresponding to the translation of the geometric primitives in characters and it automatically generates the Kasami table for that string. This table size obviously depends on the stringth length. Strings belong to the grammar if S can be found in the last row, first column of the Kasami table.

Used grammar is a context-free one and it is suitable for the recognition of different size equilateral triangles:

$S \rightarrow aA_1C$	$A_1 \rightarrow aA_2C$	$B_2 \rightarrow bB_1$
$A_1 \rightarrow b$	$A_2 \rightarrow aB_3C$	$B_1 \rightarrow b$
$A_1 \rightarrow aB_2C$	$B_3 \rightarrow bB_2$	$C \rightarrow c$

The terminal values a, b and c represent the following primitives :



Such grammar reduced to the Chomsky Normal Form can be defined as follows :

$S \rightarrow A_3A_4$	$A_4 \rightarrow A_1C$	$A_3 \rightarrow a$	
$A_1 \rightarrow A_3A_5$	$A_5 \rightarrow A_2C$	$B_2 \rightarrow B_4B_1$	
$B_4 \rightarrow b$	$A_1 \rightarrow b$	B_3	$A_2 \rightarrow A_3A_6$
$A_6 \rightarrow B_3C$	$B_1 \rightarrow b$	$A_1 \rightarrow A_3A_7$	$A_7 \rightarrow B_2C$
$B_3 \rightarrow B_4B_2$	$C \rightarrow c$		

It has been verified that the program can correctly label as 'equilateral triangles' the ones corresponding to the strings w ="*abc*" and y ="*aabbcc*"; while it labels w = "*aba*" and y ="aabbca" as 'not –equilateral triangles'.

5. Forecasts

Languages used to describe noisy and distorted patterns are often ambiguous: one string or pattern can be generated by more than one language, so patterns belonging to different classes may have the same description, but with different probabilities of occurrence (Fu,1982)

Also, a pattern grammar may generate some strings which are unwanted. Context-free grammars and also transformational grammars can represent the phrase structure of a language, however they are not able to cope with the real relative frequency or likelihood of a phrase.

The following different approaches have been proposed:

- approximation
- transformational grammars
- stochastic grammars
- similarity and error-correcting parsing

The use of approximation reduces the effect of noise and distortion at the preprocessing and primitive extraction stage.

The second approach defines the relations between the noisy pattern and the corresponding noise-free pattern by a transformational grammar.

When a nosy pattern has two or more structural descriptions (it is accepted by two or more different pattern grammars), it is proper to use stochastic grammars: to each production rule of the grammar is assigned a probability of selection, a number between one and zero. During the derivation process, productions are selected for rewriting according to their assigned probabilities: each string of the language has a probability of occurrence computed as the product of the probabilities of the rules in its derivation.

When a string has two or more parses, we can use the more probable parse as a description of the string (pattern): the most probable parse is that according to which the generated string has the highest probability.

However, what we already know about probable occurrence plays a meaningful role. The parsers are made up according to a likelihood criterion. However, parsers may also be built according to a further criterion, i.e. the Bayes's theorem.

In this case, some utter a priori information is required about the starting probability to deal with one class of patterns or another one.

When a distorted pattern cannot be accepted by any grammar, an error-correcting parsing, based upon a similarity criterion, can be used.

The above said test is the use of a normal context free grammar over an equilateral triangle, that is a well designed specimen.

We mean in the next future to test a badly designed specimen either by proper use of a normal grammar after a pre-treatment of the specimen, or using from the start

the correspondings stochastic context free grammar: a final comparison of the results will prove which is the best.

Furthermore, once stated whether to use a stochastic grammar, one should choice the grammar type.

As already said, in GIS environment, the problem is the reconnaissance of simple recursive features. Also the type of the parsing algorithm is important for correct operation.

A way of specifying a language is in terms of the strings that are accepted by a recognition device. The simplest recognizer is the finite-state automaton, which can accept languages produced by finite state grammars (Aho,1972).

A finite state automaton is a model composed of states, wich are connected by state transitions: the states correspond to the non-terminal symbols, and the state transitions correspond to the production rules of the equivalent regular (finite state) grammar.

Recognizers for 0,1 and 2 type languages and extensions (like programmed languages) have been introduced.

The theory of stochastic automata define the class of languages accepted by stochastic automata.

A stochastic finite state automaton is a five-tuple:

$$SA = (\Sigma, Q, M, \pi_0, F)$$

where Σ is a finite set of input symbols for the strings (the alphabet) , Q is a finite set of internal states, M is a mapping of Σ into the set of n×n stochastic state transition matrices, π_0 is the n-dimensional initial state distribution vector and F is a finite set of final states.

The alphabet Σ is equal to the set of terminal symbols V_T; the state set Q is the union of the set of non terminals V_N and the states T and R, state of termination and of rejection respectively; π_0 is a row vector with the component equal to 1 in the position of state S, the other components equal to 0; the state transition matrices M are formed on the basis of the stochastic productions; finally a n vector π_f represents the final state.

Indeed the generation process from a stochastic finite state grammar can be assimilated to a finite state Markov process: Hidden Markov Models (HMMs) and stochastic regular grammars are equivalent.

Stochastic context free grammars are more powerful in order to describe languages: additional rules allow to create nested, long-distance pairwise correlations between terminal symbols.

Since context sensitive grammars prove uneasy to handle, as too complex, we shall compare the first two types, finite state and context free grammars, to check their cost in terms of difficulty and computation time.

A variety of Cocke-Younger-Kasami (CYK) algorithm allows to find out an optimal parse tree (alignment problem) for stochastic context free grammars in Chomsky Normal Form; the so called "inside algorithm" allows to find out probability of a given sequence (string) if the grammar is previously known.

The inside algorithm can be compared with the forward algorithm for HMMs, the same as the CYK algorithm can be compared with the Viterbi algorithm used for HMMs.

References

Aho, A., Ullman, J., 1972. The theory of Parsing Translation, and Compiling. Prentice Hall, Englewood Cliffs.

Chomsky, N., 1957. Syntactic structures. Mouton, The Hague

Fu, K., 1974. Syntactic Methods in Pattern Recognition. Academic Press, New York

Markov, A., 1913. An example of statistical investigation in the text of "Eugene Onegin". Proceedings of the Academy of Sciences of St. Petersburg

Sester, M., 1992. Automatic model acquisition by Learning. IAPRS Vol. XXIX, Part B3.