VISUAL BASIC APPLICATION OF THE EARLEY ALGORITHM

Zeynep ALTAN
zaltan@istanbul.edu.tr
Istanbul University, Faculty of Engineering,
Department of Computer Science,
34850 Avcılar, Istanbul

Key Words: Recognition and Parsing, Earley Algorithm, Computational Linguistics, Formal Language Theory.

ABSTRACT

As abstract systems have become more sophisticated, natural language processing systems have been one of the most interesting topics of computer science. Because of the contributions of Turkish to computational applications and the language's rich linguistic properties, Turkish studies are approved in linguistic theory. This study presents a Visual Basic application of the Earley Algorithm that parses the sentences being independent from the language in a visual environment.

I. INTRODUCTION

The purpose of the abstract systems has been the simulation of the words or the sentences to obtain the speech recognition algorithms. Context-Free Grammars (CFGs) which are widely used to parse the natural language syntax are the fundamental grammars of these systems. Although other types of the grammars in Chomsky hierarchy[1] are fairly powerful, they have some disadvantages during the modelling. For example; it may not formally be possible to model the syntax of sentences by context-sensitive grammar. Because, time complexity problem may arise while the algorithm parses a sentence. Therefore, it is required to choose the grammars with less time complexity instead of the
grammars that parse the text more effectively than others.

The Earley Algorithm, which was constituted by Jay Earley as his Ph. degree thesis, has also been built by using context-free grammar *[6]*. A lot of artificial intelligence researchers have been making use of this algorithm in their studies, which are about speech recognition. The Earley Algorithm's top-down control structure depends on both CKY parsing and Knuth's LR (k) algorithm (bottom-up parsing methods).[2] This algorithm has some advantages according to other context-free based algorithms. For example; Knuth's LR(k) algorithm can only work on some subclass of grammars, so they can be done in the time n and, they are called as restricted algorithms including a lot of ambiguities. CKY algorithm parses any string with the length n in time proportional to $O(n^3)$ *[7]*. The time complexity of the Earley Algorithm for CFGs also depends on a number of special classes of grammars. If the parsing steps are defined according to an unambiguous grammar, the processes execute in $O(n^2)$ reasonably defined elementary operations, but for ambiguous context-free grammars it is required $O(n^3)$ elementary operations, when the length of the input is n. Another advantage of the Earley Algorithm is the definition of the grammar. CFG grammar used by this algorithm does not require to be defined in the Chomsky Normal Form (CNF) [3].

Pitsch presented a generalisation of the context-free LL (k) notion onto coupled context-free grammars by constituting the steps of the predictive context-free parsing machine according to Earley parser *[5]*. Stolcke defined an extension of Earley's parser for stochastic context-free grammars computing the prefix and substring probabilities, which are suitable for the original Earley chart structure *[1]*. Thus, the probable parses of substrings can be ruled out by

[1] Right Linear, Context-Free, Context-Sensitive and Unrestricted Grammars define the Chomsky hierarchy of grammars.

[2] CKY (Cocke -Kasami-Younger) Algorithm is a simple procedure for recognising strings in a context-free language, which is in Chomsky Normal Form; thus the derivation tree of any string will essentially be binary.
Knuth's Algorithm works on LR (k) grammars; i.e. rightmost derivations of sentences are obtained.

[3] A context-free grammar G = (N,Σ,P,S) is said to be in Chomsky Normal Form, if every rule is in one of the following forms:
$X \rightarrow YZ$, $X \rightarrow a$ for $X,Y,Z \in N$ and $a \in \Sigma$.

the top-down modelling. Briscoe and Carrol developed an interactive incremental parsing system constructing the LALR (1) parse table defined by ANTL (The Alvey Natural Language Tools) grammar. This system includes lexical, morphological and syntactic analysis of English *[8]*.

Most of the recognition algorithms that depend on the formalism of tree-adjoining grammar (TAG)[4] use the steps of Earley Algorithm to parse the sentences according to the compiled grammar. Schabes and Schieber studied the extended derivation of TAGs with the application of Earley Algorithm deducting the set of Earley items on the corresponding grammar *[9]*. Minnen developed the predictive left-to-right parsing of the restricted TAG(LD/LP) (local dominance/linear precedence) with an algorithm, which was closely related to the Earley parsing *[4]*. Thus the schematic representation of trees and the combination of these trees with the adjunction operations could allow to the various permutation of the elementary structures.

Because of the contributions of Turkish to computational applications and the language's rich linguistic properties, Turkish studies are approved in linguistic theory. This study presents a Visual Basic application of the Earley Algorithm parsing the sentences in a visual environment. Since the tool is independent from the language, we can define grammar rules both for Turkish and English. Our next study will focus on the extension of this algorithm for TAGs constituting a similar recogniser, so the advantage of TAGs according to CFGs will be adapted to the recent application. We are planning to test the TAG(LD/LP) recognition algorithm which LD/LP are defined as constraints and structure, respectively[5]. After we described the grammar rules in Turkish according to morphological properties of the inflected words *[10],* the most important disadvantage of using Early Algorithm has been eliminated. In this way, the declaration of all words in the input sentence does not require; since the root words are saved in the database, the number of the grammar rules including terminal categories will reduce and become general containing only suffix rules. Verb inflections in Turkish may also be defined by grammar rules. The verb in the input sentence, which precedes the suffixes, is analyzed as an invariant root by querying the database, and the following suffix particles may indicate voice (causative, reciprocal, reflexive, passive), modality (necessitive, abilitative, conditional), negation, tense-aspect mood and person/number. This property also reduces the number of rules defined for terminal categories including verbs. As a result, morphological analysis is very meaningful for the determination of part-of-speech structure in syntactic parsing, and for the semantic analysis of a sentence. Information about verbal inflection is especially important for the word order concept *[11]*.

II. TOP-DOWN APPROACH OF THE EARLEY ALGORITHM

Any context-free rule format can be adapted to the Earley Algorithm to parse a string or a sentence with the productions of given grammar building the left-most derivation of the strings *[2]*.

Let $E_{i,j}$ be any state in the state set which is derived from a consistent production. Then it can be represented as:

$$E_{i,j}: A \rightarrow \alpha . \beta ,$$

where i is the initial position of any nonterminal A which is expanded to supply the condition $i \leq j \leq n$, $i < n$ (n is the last symbol of input string) and, j is the current state of which any string in the form $x = w_1 w_2 \ldots w_{j-1}$ begins to process.

The expansion of A is repeated until the preceding sentential form is completed to yield a derivation form $x = w_1 w_2 \ldots w_n$. Any production of the grammar gives a left most derivation as:

$S \Rightarrow^* w_1 \ldots w_i A\delta \Rightarrow^* w_1 \ldots w_i \alpha\beta\delta \Rightarrow^* w_1 \ldots w_j \beta\delta$ *[3]*.

Thus the dotted production $A \rightarrow \alpha . \beta$ (either α or β may be empty) is in $E_{i,j}$. Each state in the Earley Algorithm represents the following components:

(a) production, which is derived from the right of input string scanning a part of $x = w_1 w_2 \ldots w_n$;

[4] If the sentences to be parsed are generated as small pieces, which are called elementary trees out of the phrase structure, tree-adjoining grammar (TAG) is defined as formalism. Then these small pieces with some constrained conditions are composed to form larger pieces of tree structure.

[5] The use of the top-down approximation of the Early Algorithm can form a large number of unnecessary items to be predicted and unsuccessful intermediate results can be obtained when the grammar size to be parsed is too large. But the basic idea of parsing with TAG structure reduces these unnecessary predictions with the adjunction operations. The adjunctions for all derivations are eliminated to create new relations between the supertrees of the roots and the subtrees of the foots.

(b) a point that shows which part of the production's right side has been recognised so far;
(c) a pointer, back to the position looking for the production in the input string ;
(d) a lookahead (k-symbol) string, which can be used instead of successive production.

In this application of the Earley Algorithm, the lookahead string, which gives a property of Earley states, has been neglected and a matrix form with two indices has been used as the pointer.

III. FORMAL EXPLANATION OF THE EARLEY ALGORITHM

Let G = (N,Σ, S, P) be a CFG without containing Λ-productions. If the input string is given as $x = w_1 w_2 \ldots w_n \in \Sigma^*$, then the states $E_{i,j}$ with the conditions $0 \le i \le j \le n$ and $i<n$ are calculated as follows:

(a)Initialisation: For $\forall P: S \rightarrow \gamma$;
place $E_{0,0}$: $S \rightarrow .\gamma$

Step (b) is repeated until no new element is added to $E_{0,0}$.

(b)Prediction: For $\forall A \in N \quad B \rightarrow . A \beta$ is in $E_{0,0}$
and, for $\forall P : A \rightarrow \gamma$;
$A \rightarrow . \gamma$ is added to $E_{0,0}$

Then for $\forall j>0$ and $\forall i,k$, after the construction of $E_{i,k;}$ the steps (c), (d) and (e) are repeated until no dotted productions are added to $E_{0,j}$, $E_{1,j} \ldots, E_{j,j}$

(c)Scanning : If $E_{i,j-1} : A \rightarrow \alpha . x_j \beta$;
the state $A \rightarrow \alpha x_j . \beta$ is added to $E_{i,j}$

(d) Prediction: For $A \in N$, if $B \rightarrow \alpha .A\beta$ is in $E_{i,j}$;
for $\forall P: A \rightarrow \gamma$, $A \rightarrow .\gamma$ is added to $E_{j,j}$

(e) Completion: If $E_{i,j} : A \rightarrow x.$ is the completion state and $B \rightarrow \alpha . A\beta$ is in $E_{k,i}$;
$B \rightarrow \alpha A .\beta$ is added to $E_{k,j}$

IV. PARSING WITH NEW TOOL

When the parser is run, the project file called "earley.vbp" is opened including three different form-modules "about.frm", "earley.frm" and "print.frm" in it. Generally form files of a project are saved in the project directory. The file named "about.frm" includes the definition of the control objects in the package. Command, frame, label and picture are the objects of this form and each of them defines different procedures. "earley.frm" includes the codes of the algorithm and the general object constituting all of these codes. The other objects in this form represent the codes of menu elements. The third form-module, "print.frm", organises the outputs.

Visual Basic is extremely flexible in designing the user interfaces and makes possible to add user interface components. We can add these elements, for example; text boxes, dialog boxes, list boxes and sign boxes, by using control devices. One of the advantages of programming with Visual Basic is the speed in developing and testing of an application. Before testing, it is not required a condition that the application must be finished. When a new property is added to the application, this property is tested; if we change something on it, this change can also be tested again.

The application of the Earley Algorithm using this parser tests many different sentences both for Turkish and English. After the sentences and the grammar rules describing these sentences have been defined, the parsing procedures of given examples are completed successfully. The basic purpose of the constituted parser is the running of the algorithm as fast as possible. If it is required to use this parser for any text recognising problem, the agreement of each sentence with previously defined grammars is tested, and then the parsing process is executed. If the input sentence fits none of the grammar rules, new grammar rules for this sentence are defined. Different screen outputs of the analysis results for English sentences can be seen in the Appendix.

V. CONCLUSION

In this study, we present an application of the Earley Algorithm by using Visual Basic programming language. The most important property of this structure is the generation of the left most derivation tree from top-down and left to right, and the main purpose of the algorithm is to recognise the sentences in different languages according to their semantic and lexical features.

During the execution, in case of the complication for the next action of the parser, it is ambiguous which production must be chosen. This nondeterministic situation increases the number of searched processes as unsuccessful intermediate results of the Earley analysis. As the applications use large grammars, Earley recognition steps will include a lot of ineffective processes. It is possible to eliminate this negation defining a deterministic top-down strategy by using a context-free LL(k) parsing algorithm.

Syntactic structures of the sentences are also an important collapse of the Earley Algorithm. This technique doesn't include transformational grammar which contains two components, one

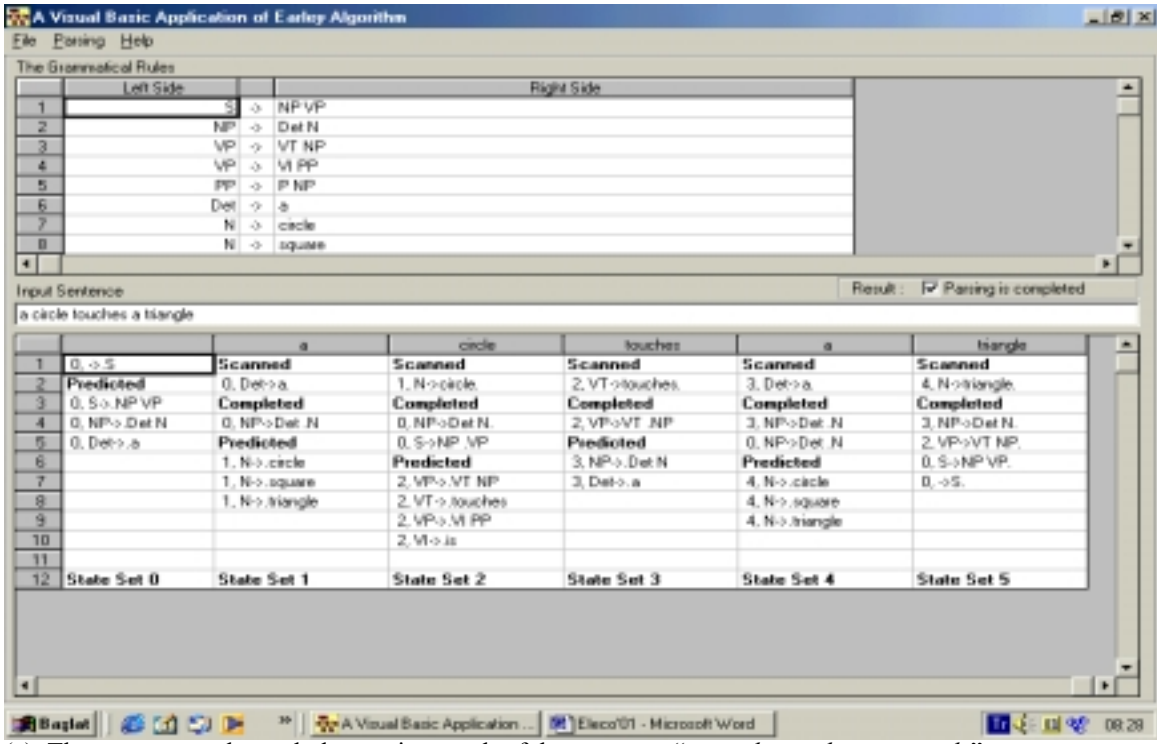
of which is called the base component and the other, transformational component. But we eliminated this problem by analysing the morphological properties of the words. We can define transformational grammar rules including inflectional suffixes of Turkish words. Traditionally, the analysis of word structure is divided into two basic fields as inflection and derivation. Therefore, the morphological structure of each word may include elements such as prefix, suffix, infix, or even a separate root, and these elements can modify the meaning of the basic root or stem of the word. The tool that we developed can analyze all the sentence structures in case of correct definitions.

If we don't define the rules according to the morphological properties of the words, sentences vary with respect to the tense (present or past), the number (singular or plural), and the aspect (question, negation, passive, active or statement). It is also difficult to answer the following question: How can anyone release that all of these sentences are similar, even though there are differences between their forms and meanings? In this case, the grammar rules of the algorithm include lexical components for terminal categories as N→{chicken, boys, girl, house, table}, Det→{a, the, an}, V→{ate, borrow, gives}.

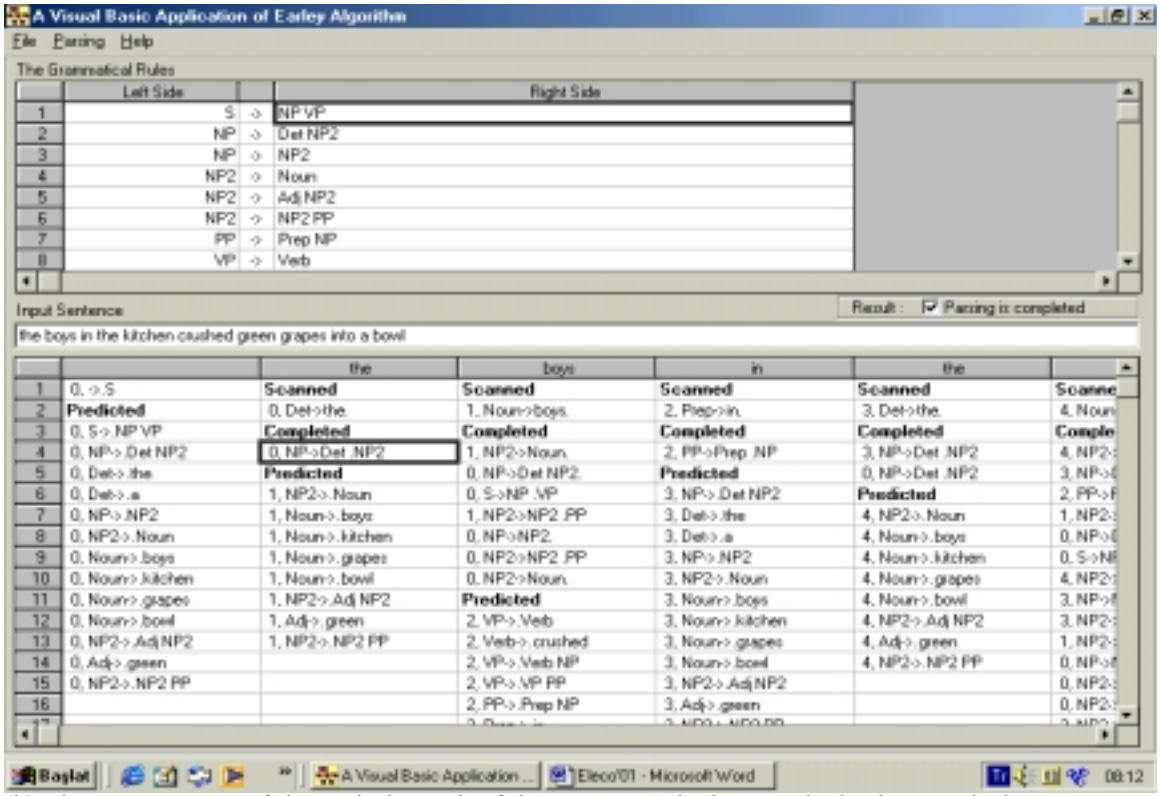
REFERENCES

1. A. Stolcke. (1995)."An Efficient Probabilistic Context-Free Parsing Algorithm that Computes Prefix Probabilities" *Assoc. for Computational Linguistics*, Vol. 21, pp. 165-200.
2. A.V. Aho and J.D. Ullman. (1972). *The Theory of Parsing*, *Translation and Compiling*. Prentice Hall, New Jersey.
3. G.K. Krulee. (1991). *Computer Processing of Natural Language*, Prentice Hall, New Jersey.
4. G. Minnen. (1994). "Predictive Left-to-Right Parsing of a Restricted Variant of TAG(LD/LP)", In *Computer Intelligence,* Vol. 10, Num. 4, pp. 535-548.
5. G. Pitsch. "LL (k) Parsing of Coupled-Complex -Free Grammars", In *Computer Intelligence*, Vol. 10, Num 4, pp.563-578.
6. J. Earley. (1986) "An Efficient Context-Free Parsing Algorithm", B J.Grozs, K.S.Jones and B.L.Webber (eds.), In *Readings in Natural Language Processing,* pp.25-33
7. R.N. Moll, M.A. Arbib, and A.J. Kfoury. (1988). *An Introduction to Formal Language Theory,* Spring-Verlag, New York.
8. T. Briscoe and J. Carrol. (1993)"Generalised Probabilistic LR Parsing of Natural Language with Unification-Based Grammars", In *Assoc. for Computational Linguistics*, Vol. 19, Num.1, pp.25-58.
9. Y. Schabes and S. M. Shieber. 1994. "An Alternative Conception of Tree-Adjoining Derivation", In *Assoc. for Computation Linguistics*, Vol. 20, Num. 1, pp. 91-124.
10. Z.Altan and K. Aydın . (2000). İsimlerde Çekim Eklerinin Oluşturduğu Ses Olaylarının Visual-Basic Ortamında İncelenmesi. In *Elektrik-Elektronik-Bilgisayar Mühendisliği Sempozyumu*, Bursa, Turkey, pp. 307-312.
11. Z.Altan. (2001, in print) "The Role Of Morphological Analysis In Natural Language Processing" In *Anadolu University Journal of Science and Technology*, ISSN 1302-3160.

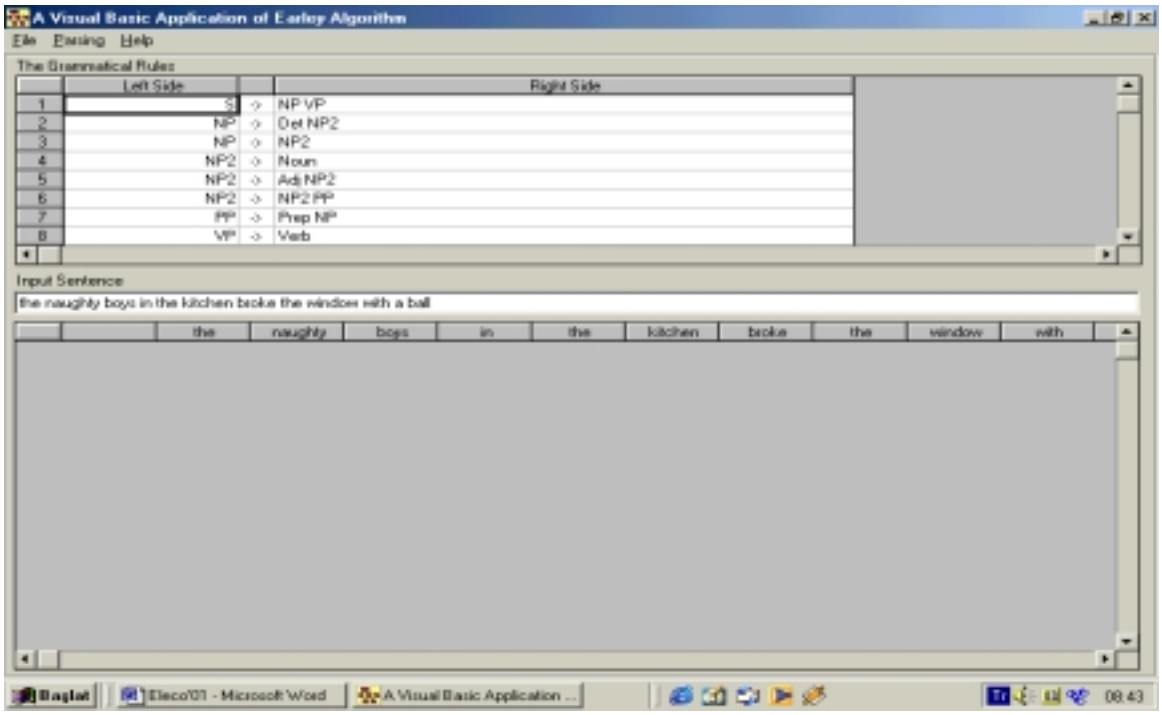
APPENDIX



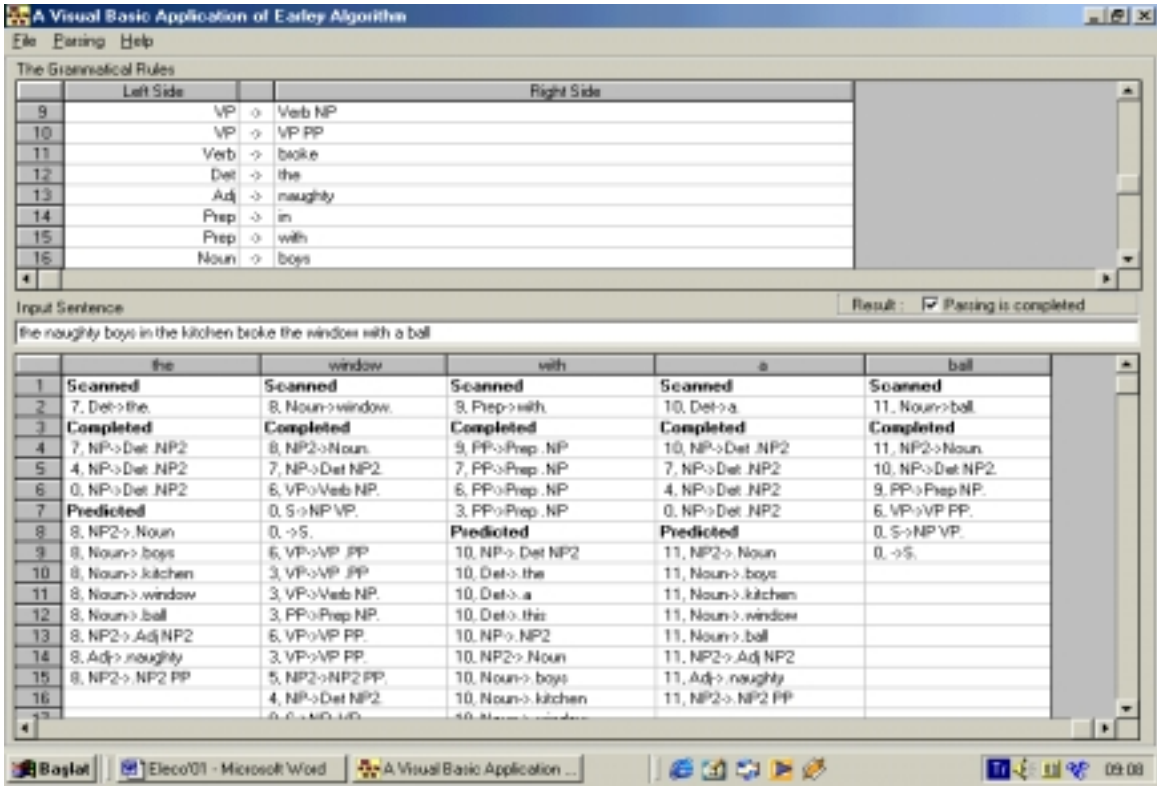
(a) The grammar rules and the parsing result of the sentence *" a circle touches a triangle*"



(b) The screen output of the analysis result of the sentence*" the boys in the kitchen crushed green grapes into a bowl*"



(c) The interface before the parsing process began for the sentence "*the naughty boys in the kitchen broke the window with a ball"*. It is also possible to parse the sentence, which has no meaning *"the yellow book on the table dropped the chair from the wall"* with the same grammar rules.



(d) The interface after the parsing has been completed for the sentence *"the naughty boys in the kitchen broke the window with a ball"*.