NATIONAL CENTER FOR SCIENTIFIC RESEARCH
"DEMOKRITOS"

Institute of Informatics
& Telecommunications

Software & Knowledge Engineering Laboratory

# *Ellogon*
# Components' Specifications

Stergos D. Afantenos
George Petasis
Vangelis Karkaletsis

*June 2002*

# Table of Contents

Chapter 1

# 1 Introduction

This document is intended to accompany the *Ellogon* text engineering platform, along with two other documents: the *Users' Guide to Ellogon* and the *Developers' Guide to Ellogon*. In those manuals we have described the *Ellogon* platform in detail, and we have furthermore described how to build and run Components and Systems for *Ellogon*, among other things.

The current distribution of *Ellogon* comes equipped with some predefined Components. The purpose of this document is to describe those Components and what they are doing.

Chapter 2

# 2 Greek Tokenizer Splitter

The aim of the Greek Tokenizer Splitter Component is to identify all the tokens inside a text and annotate them properly. Tokens are all the words which are separated between them with blank spaces or punctuation marks. Additionally, all the punctuation marks are considered to be tokens.

Furthermore, this Component also tries to identify sentence boundaries. In order to achieve this, it uses a simple algorithm based mainly on punctuation marks and capitalization.

## 2.1 Input

The input to the Greek Tokenizer Splitter is just a Collection of Documents which need not have any other annotations, from another Component. In other words, this module has no pre-conditions. The only constraint that this component poses is for the Documents to be written in the Greek language.

## 2.2 Output

The Greek Tokenizer Splitter tries to identify all tokens of a Document and annotate them appropriately according to their type. By the type of a token we mean an indication on whether it is, for example, a punctuation mark, an English uppercase word, a Greek lower case word, *etc.* In other words, this Component places annotations of type `token` for all the token it identifies with an attribute `type` according to the type of the token. You can see all the types of attributes that this Component places in *Table 2.1*. In *Figure 2.1* you can see an example of a token annotation.

This Component also tries to identify all the sentences in the Document. In order to do so, it uses a simple algorithm which is based on punctuation marks and capitalizations. For every sentence it finds it creates a new annotation of type `sentence` with attribute `constituents`. The constituents contain the ids of all the tokens which constitute the sentence. An example of a sentence annotation you can see in *Figure 2.2*.

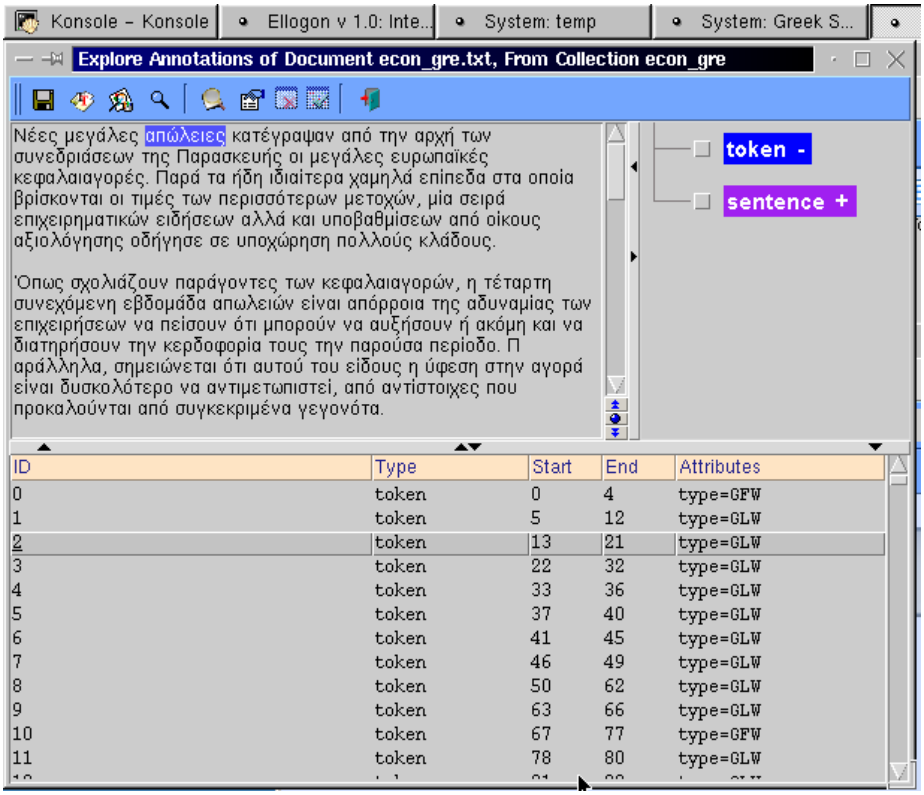


*Figure 2.1An Example of a Token Annotation*

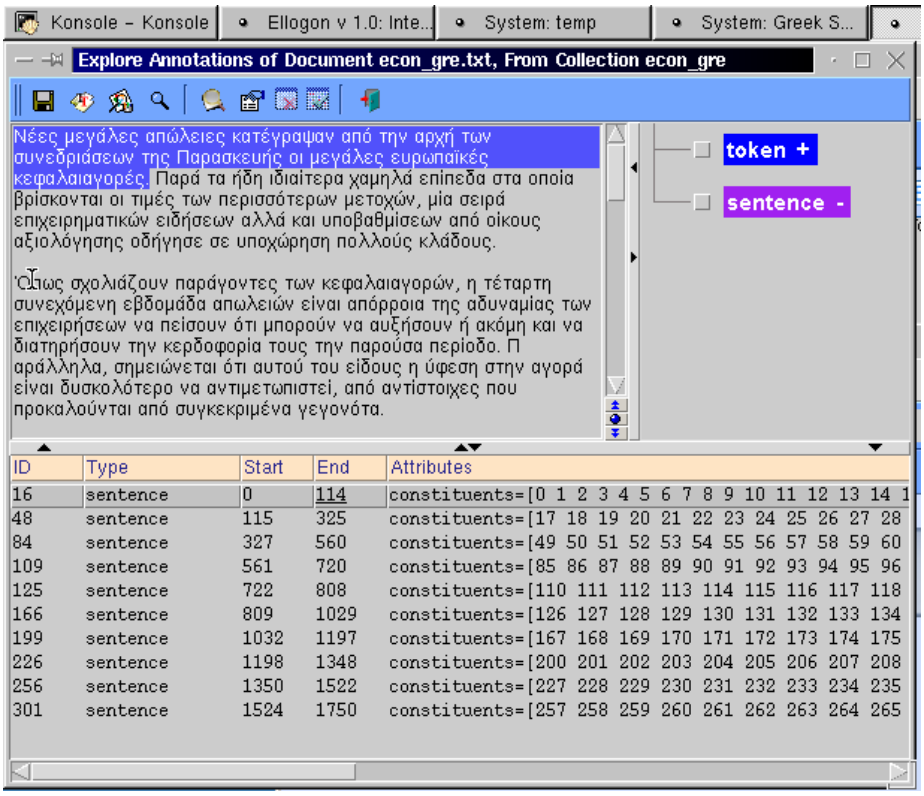


*Figure 2.2 An Example of Sentence Annotation*

| *Token Type* | *Token Description* |
|---|---|
| GLW | Only Greek lower case characters are present in the token (α-ω, e.g. "λέξη"). |
| GUW | Only Greek upper case characters are present in the token (Α-Ω, e.g. "ΛΕΞΗ"). |
| GFW | Only Greek characters comprise the token. The first is an upper case character and the rest are lower case characters (e.g. "Λέξη"). |
| GW | Only Greek characters are present in the token and the token does not belong to the GLW, or GUW, or GFW types (e.g. "λΕξΗ"). |
| ELW | Only lower case, Latin characters are present in the token (a-z, e.g. "word"). |
| EUW | Only upper case Latin characters comprise the token (A-Z, e.g. "WORD"). |
| EFW | Only Latin characters comprise the token. The first character is upper case and the rest are lower case (e.g. "Word"). |
| EW | The token is comprised of Latin characters only and it does not belong to any of the ELW, or EUW or EFW types (e.g. "wOrD"). |
| MLW | The token is comprised of Greek lower case and Latin lower case characters. It does not contain characters of any other kind (e.g. "καπατel"). |
| MUW | The token is comprised of Greek upper case characters and Latin uppercase characters. The token does not contain characters of any other kind (e.g. "ΚΑΠΑΤΕL"). |
| MFW | The token is comprised of Greek and Latin alphabetic characters. It does not contain characters of any other kind. The first character is uppercase and the rest are lower case (e.g. "Καπατel"). |
| MW | The token is comprised of Greek and Latin characters and it is not classified as MLW or MUW or MFW (e.g. "καπαTEL"). |
| NUM | The token is comprised of numeric characters exclusively (0-9, e.g. "1999"). |
| WNUM | The token is comprised of one or more numeric characters and Greek or Latin or both alphabetic characters The token does not belong to any of the aforementioned categories (e.g A1). |
| PUNCTUATION | The token is one of the following characters: ! . , ? ; :... |
| SYMBOL | The token does not belong to any of the aforementioned categories. |

*Table 2.1Types of Attributes for the Tokens*

Chapter

3

# 3 Hellenic POS Tagger

The Hellenic POS Tagger tries to identify the part of speech for every token in the Document. This component is based on code written by Eric Brill in 1995[1] and was later modified by George Petasis in 1999[2] in order to find Greek parts of speech.

## 3.1 Input

The input to this Component are Documents written in the Greek language. Also this component presupposes that the Greek Tokenizer Splitter Component has been run against the Collection. In other words it expects to find `token` and `sentence` Annotations in the Documents.

## 3.2 Output

The result of the processing of this Component is simply to add an additional attribute to every token. The attribute is of type `pos` and it indicates what part of speech this token is. If, for some reason, it fails to find a part of speech for a specific token, then the attribute `pos` is empty. An example you can see in *Figure 3.1*. In the Table of the following page you can see all the possible values of the pos attribute.

[1] Brill, E., "Transformation-Based Error-Driven Learning and Natural Language Processing: A CaseStudy in Part of Speech Tagging", Computational Linguistics, vol. 21, n. 24, 1995.

[2] G. Petasis, G. Paliouras, V. Karkaletsis, C.D. Spyropoulos and I. Androutsopoulos, "Using Machine Learning Techniques for Part-of-Speech Tagging in the Greek Language", *Proceedings of the 7th Hellenic Conference on Informatics*, Ioannina, Greece, 1999.

# Categories

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | **Articles** | | | | | | |
| DDT | Definite Article | | | IDT | Indefinite Article | | |
| | **Nouns** | | | | | | |
| | **PoS** | **Number** | **Gender** | | **Pos** | **Number** | **Gender** |
| NNM | Noun | Singular | Male | NNPM | Proper Name | Singular | Male |
| NNF | Noun | Singular | Female | NNPF | Proper Name | Singular | Female |
| NNN | Noun | Singular | Neuter | NNPN | Proper Name | Singular | Neuter |
| NNSM | Noun | Plural | Male | NNPSM | Proper Name | Plural | Male |
| NNSF | Noun | Plural | Female | NNPSF | Proper Name | Plural | Female |
| NNSN | Noun | Plural | Neuter | NNPSN | Proper Name | Plural | Neuter |
| | **Adjectives** | | | | | | |
| | **PoS** | **Number** | **Gender** | | **Pos** | **Number** | **Gender** |
| JJM | Adjective | Singular | Male | JJSM | Adjective | Plural | Male |
| JJF | Adjective | Singular | Female | JJSF | Adjective | Plural | Female |
| JJN | Adjective | Singular | Neuter | JJSN | Adjective | Plural | Neuter |
| CD | Cardinal Numerals (e.g. ἑνα (one), δύο (two), τρία (three), ... and numbers: 1, 2, 3...) | | | | | | |
| | **Pronouns** | | | | | | |
| PRP | Personal Pronoun | | | IP | Demonstrative Pronoun | | |
| PP | Possessive Pronoun | | | WP | Relative Pronoun | | |
| REP | Reflexive Pronoun | | | QP | Interrogative Pronoun | | |
| DP | Definite Pronoun | | | INP | Indefinite Pronoun | | |
| | **Verbs** | | | | | | |
| VB | Present Tense Verb | | | VBS | Present Tense Verb, Plural | | |
| VBD | Past Tense Verb | | | VBDS | Past Tense Verb, Plural | | |
| VBF | Future Tense Verb | | | VBFS | Future Tense Verb, Plural | | |
| MD | Modal Verb (ἐχω (have), εἰμαι (be)) | | | | | | |
| | **Participles** | | | | | | |
| VBG | Active Voice Particle | | | VBPD | Past Participle | | |
| VBP | Present Participle | | | | | | |
| | **Ἀκλιτα Μέρη του Λόγου** | | | | | | |
| RB | Adverb | | | RP | Particle | | |
| IN | Preposition | | | UH | Expletive/Exclamation | | |
| CC | Conjunction | | | FW | Non-Greek word | | |
| | **Other Symbols** | | | | | | |
| DATE | Date | | | : | Colon (:) | | |
| TIME | Time | | | ; | Semi Colon (i.e. Greek Question Mark) (; ?) | | |
| AB | Abbreviation | | | ! | Exclamation Mark (!) | | |
| SYM | Symbol | | | ( | Left Bracket "(" | | |
| . | Full stop (.) | | | ) | Right Bracket ")" | | |
| , | Comma (,) | | | " | Left Quotation Mark ( ", «) and Right Quotation Mark (", ») | | |

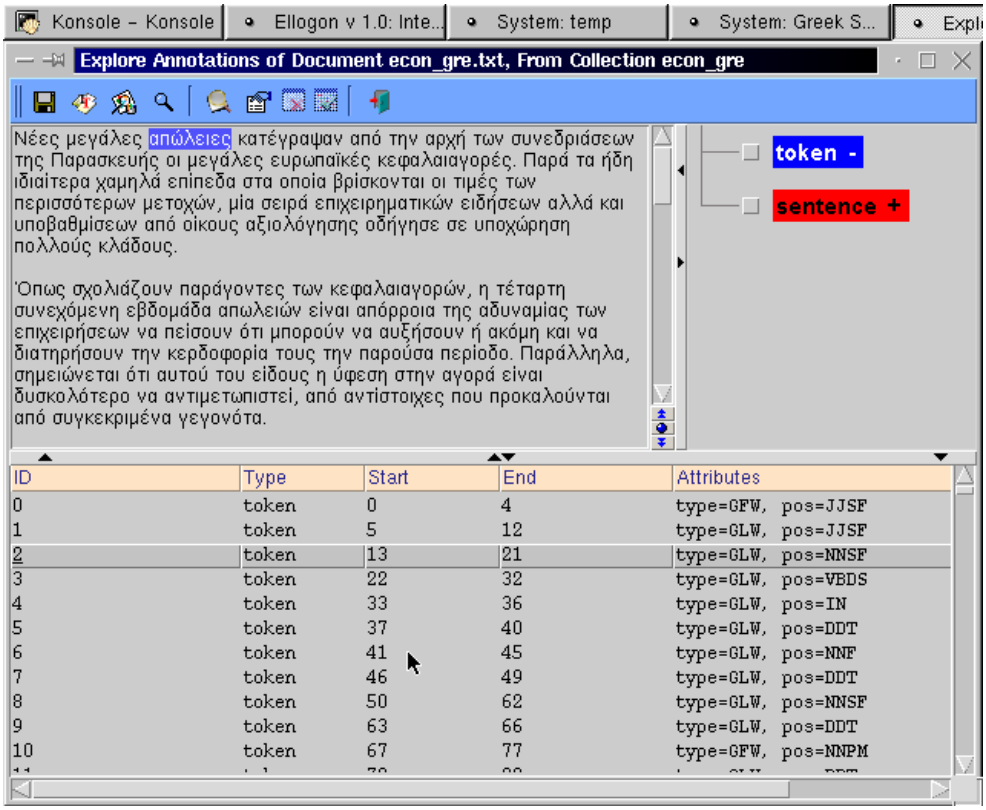


*Figure 3.1An Example of a Part of Speech Attribute*

Chapter 4

# 4 Unicode List Lookup

The function of the Unicode List Lookup Component is to compare the tokens (see Chapter 2) in the Documents of a Collection against a set of lists. For each of the elements in the lists if it matches a token, a new annotation is created, with the same spans as the token. The type of the annotation is lookup and the attribute is the name of the file which constitutes the list.

This Component is especially helpful, for example, in cases you have some lists with named entities, such as persons' names, locations' names, organizations' names, *etc.*

## 4.1 Input

The input to this component is a Collection which has run against a tokenizer, for example the Greek Tokenizer splitter described in Chapter 2. Additionally, this component needs a set of lists which you can give as input through the parameters of the Component. Note that in the present distribution of Ellogon, there are no lists and the Ellogon user should develop his own. In *Figure 4.1* you can see the parameters of the Component. in order to add your own lists push the "Create Rule File". Additionally, in this dialog you can specify the encoding of your list files, and the rule file (see below).

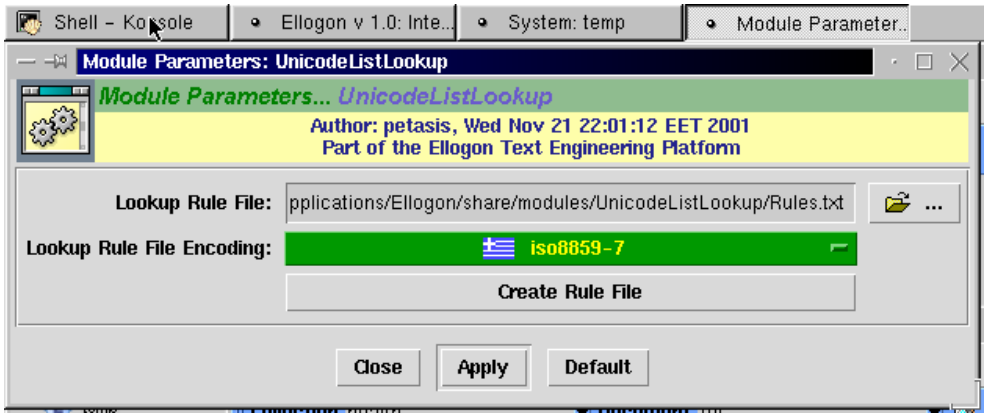


*Figure 4.1 The Unicode List Lookup Module's Parameters*

In *Figure 4.2* you can see the dialog that appears. With this dialog you can specify the list files that you want by pressing the associated Browse button. Note that the name of your list files should have the extension `.lst`. Also, each line of your list file should contain only one element.

Apart from the list files, you can also define a name for the rule file. This rule file will be created by the Component, and will be based on the list files you have specified. *Ellogon* provides a default name for that file.

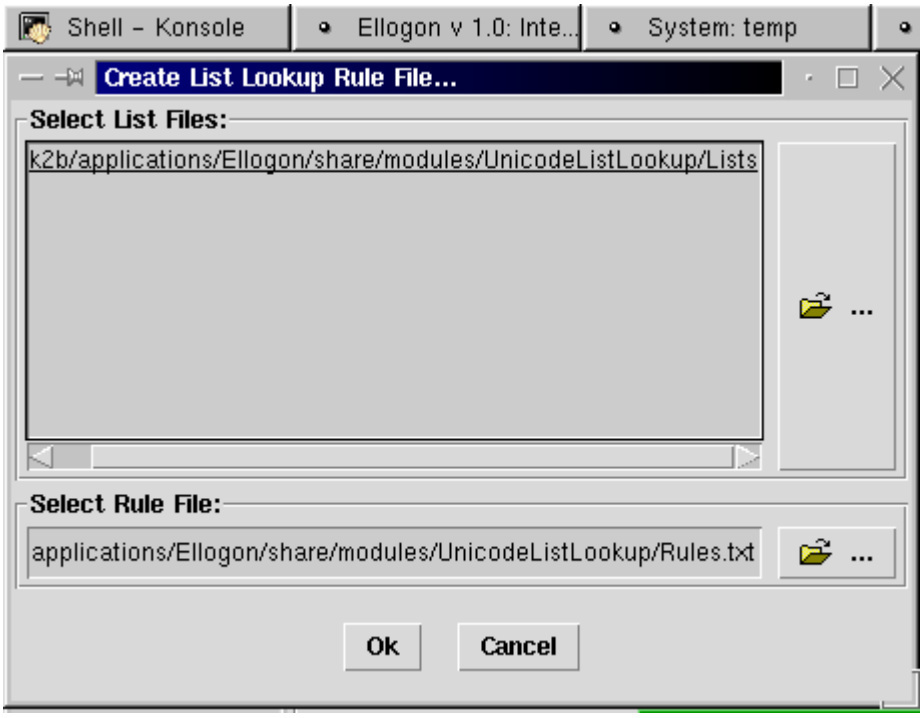


*Figure 4.2 Defining your own lists*

Once you have finished with defining the list files push the OK button

## 4.2 Output

This Component tries to match the elements of all the lists with the tokens of the Documents in the Collection. For every token that matches an element of a list with the name `my_list.lst` it creates a new annotation of type `lookup` and attribute `my_list`.

Chapter 5

# 5 Add Sentence Constituents

The function of the Add Sentence Constituents Component is to calculate the constituents, *i.e.* the ids of the tokens that constitute a sentence, and place them as an attribute in the sentence annotation.

## 5.1 Input

The Add Sentence Constituents Component takes as input Documents in a Collection which contain sentence annotations. Usually those annotations will also have as attribute a set of constituents which contain the ids of the tokens which constitute the sentence.

## 5.2 Output

This component discards all the constituents of the sentence annotation and recalculates its constituents. Then it places the ids of the tokens that it has calculated that constitute the sentence as the new constituents attribute.

This Component is especially useful in cases where some tokens have been deleted from the Documents. In such a case you can "correct" the sentence constituents with this Component.