Abstract. This paper describes the syntactic engine for the Lithuanian language which is currently being developed at the Vytautas Magnus University. The program was written in Haskell and consists of four modules: a constituent analysis, a shallow constituent analysis, a dependency analysis and a dependency analysis augmented with thematic roles. The “work-in-progress” status of the project is emphasized, as well as the significant results achieved in the current stage.

Keywords. Syntactic analysis, dependency syntax, constituency syntax, thematic roles, Lithuanian language

Introduction

For several years, the development of tools designed to process the Lithuanian language has been limited mostly to morphology, while other neighbouring languages have been progressing in developing syntactic parsers (e.g. [1], [2]). The goal of this paper is to summarise the development of the current prototype of a rich Lithuanian syntactic engine, which is included in a broad web service as part of the project Syntactic and Semantic Analysis System of the Lithuanian Language for Corpus, Internet, and Public Sector. The system should be available online in a short period of time.

This engine, which was developed in Haskell, reads morphologically annotated files and produces a set of different syntactic results. Besides the core constituency analysis, the engine is able to decompose sentences into clauses and clusters (or syntagms), to produce dependency trees and to annotate dependency trees with thematic roles. This paper gives a short overview of the whole system (section 1) and then focuses on the four different components (sections 2 to 5).

1. Overview of the Engine

The engine is built around a core constituency parser serving three components, which are, respectively, “shallow”, “dependency” and “θ-role” components, all coded in Haskell and functionally combined in a pipeline architecture, shown in Figure 1.

---

1 Corresponding Author: Francesco Zamblera, KLC-VDU, K. Donelaicio g. 52-206, LT-44244 Kaunas, Lithuania; E-mail: f.zamblera@uki.vdu.lt.
The only real parser is the constituency parser, whose input is made of morphologically annotated (and disambiguated) sentences, while the other components carry out their tasks on the basis of previously parsed data, so that, properly speaking, there is no dependency and shallow parsing, but different operations on one constituency tree.

The main advantage of this architecture is that constituency trees, dependency trees and clusters/clauses are all based on a single grammar, which greatly helps to maintain the system. The thematic-role component, in its turn, analyses dependency trees to add the relevant information, and is more loosely integrated in the system.

The system is mainly intended to be integrated in a web service. That is why, to ensure the efficient flow of data between components of the system, the JSON format is being used for input and output. However, the system can also read files generated by the Lithuanian tagger, which is based on Lemuoklis [3] with some improvements such as disambiguation. Besides, dependency trees can be represented in PML (Prague Markup Language), a specification of XML. Through the PML standard, dependency trees can be visualised and modified with the tree editor TrEd [4].

2. The Core Constituency Parsing

As mentioned before, the system is rule based and built around a constituency grammar. This grammar is directly coded in Haskell according to a functional approach which has to be described in a separate paper.

Like any usual formal grammar, this grammar consists of a set of nonterminal, a set of rewriting rules and a root symbol. Terminal symbols, that is words with their morphological annotation, are not explicitly listed. Besides, this grammar has two extensions:

1. a head marking (in order to derive the dependency representation, see section 3).
2. a system of ranking which distinguishes different levels of priority among the rewriting rules (main, secondary and rare rules). Such a design allows improving performance and avoids overgeneration of syntactic analyses with a very low probability. Furthermore, this principle allows returning a single syntactic tree for a given sentence (as the best guess) instead of the set of possible trees;
Given the central role of the grammar, the selection and the ranking of the syntactic rules is carefully controlled during the development of the prototype, so that the core of the language is efficiently parsed. An example of constituency tree is given in Figure 2.

![Constituency Tree](image)

**Figure 2.** An example of a constituency tree (visualisation with Obelis)

### 3. The Shallow Parsing

The shallow parsing is directly generated from the full constituency analysis. As a consequence of the previous full analysis, the computation takes more time; the advantage, as mentioned before, is that the whole process is controlled by only one grammar. The shallow parsing extracts two kinds of syntagmatic structures:

1. clauses,
2. phrases/clusters.

Results are given as a list of coordinate clauses, possibly with nested relative or subjunctive clauses. The structure is fully recursive so that a conjunction of subjunctive clauses could be treated as a list of clauses (nested in the clause they belong to).

For the sentence *Krūmas – dažnai tavo ėgio ar kiek didesnis, ji sudaro daug sumedėjusių stiebų, kuriuos krūmai išleidžia iš kelmo.*

2 ‘A bush is often of your height or a bit taller, it is made of many woody stems, which bushes send out from their stumps’, three clauses have to be recognized: 1) *krūmas – dažnai tavo ėgio ar kiek didesnis*, ‘a bush is often of your height or a bit taller’, 2) *ji sudaro daug sumedėjusių stiebų, kuriuos krūmai išleidžia iš kelmo* ‘it is made of many woody stems, which bushes send out from their stumps’, 3) *kuriuos krūmai išleidžia iš kelmo* ‘which bushes send out from

---

2 We have chosen an example from a Lithuanian children’s encyclopedia, so that the clause structure is easy to represent.
their stumps’. The clauses 1 et 2 constitute two daughters of the whole sentence. As coordinate clauses, they are on the same hierarchical level: they constitute a list of (two) clauses, while the clause 3, the relative clause, is nested in the second clause. Formally, the segment which represents the third clause appears twice: first, as a part of the clause 2, then as the separate clause 3.

The engine recognizes different types of phrases: noun phrases (NP), prepositional phrases (PP) and verb phrases (VP). However, while Lithuanian noun phrases and prepositional phrases have a clear syntagmatic aspect, which makes them easy to treat as constituency structures, this is not the case for verb phrases. Indeed, their definition is not clear in a “free word-order language” such as Lithuanian. As a consequence, extracted verb phrases are still very problematic. For this reason, in the current stage, verb phrases are usually made of a lexical verb, possibly with its adverbial complement.

The identification of phrases depends on the clause. In the given example, the syntagm daug sumedėjusių stiebų, kuriuos krūmai išleidžia iš kelmo ‘many woody stems, which bushes send out from their stumps’ is a noun phrase of the clause 3 as a whole. But inside the clause 3, this group is divided into three different phrases: krūmai ‘bushes’ (NP), išleidžia ‘send out’ (VP) and iš kelmo ‘from their stumps’ (PP).

4. The Dependency Analysis

Dependency trees are directly generated on the basis of previously produced constituency structures. For this purpose, syntactic heads are marked in the constituency grammar: this ensures the equivalence between constituency and dependency representations [5]. Hence, dependency derivation is a trivial task, which basically involves replacing nonterminal symbols in the constituency tree with their lexical heads.

It must be emphasized that the dependency structure is a purely lexical tree (e.g., Figure 3). No information is given on the syntactic function of dependencies (e.g. subject, attribute). This choice is justified by the fact that the distinction between an object and some adverbials is directly connected with thematic role analysis. To avoid repetition of the same process, it was decided to leave this issue to the next level. This decision might be subject to future change.
5. “Thematic-role” Component

This module adds thematic roles to the nodes of the dependency tree generated by the syntactic-analysis component. It performs a head-first left-to-right recursive traversal of the dependency tree and tries to assign a thematic role to each dependent.

The list of recognized thematic roles is drawn from [6]. The module outputs Sližienė’s thematic role and the nearest VerbNet equivalent.

The main tool for thematic-role assignment is a table of verb valencies, compiled according to Sližienė’s dictionary, which includes about 2,000 entries. However, our table has about 3,000 entries because of aspectual pairs. The table stores the “thematic grid” of a verb: a list of thematic roles that a verb assigns, either obligatorily or facultatively, with morphological markers for each role (which can be a specific morphological case or a particular preposition or conjunction).

For verbs which are not in the table and for nominal constituents which are not part of the thematic frame of a verb, thematic roles are assigned mainly taking into account the case of a noun phrase and/or the preposition which governs it. In the future development, the module would ideally have access to an ontology with semantic properties (whether, for example, a noun is animate or inanimate, or expresses time or place, and so on). For the time being, only a list of time expressions has been compiled.

The main algorithm for thematic-role assignment can be sketched this way:

1. The main predicate (root node) of a sentence is identified and labelled *pred.
2. The thematic grid of the main predicate is looked for in the table; if none is found, the default one is used instead.
3. The thematic grid and the list of the immediate dependents of the main node are passed to a function which assigns each dependent its thematic role according to its morphological properties: then, if the dependent is not a terminal node, its theta-role grid is retrieved, and its dependents are assigned their thematic roles by a recursive call to the function itself.

After thematic roles are assigned, each of them is mapped onto its VerbNet equivalent. Sližienė’s list of thematic roles is long and very detailed, so its (mostly many-to-one) mapping onto VerbNet did not create any big problem. However, in a couple of cases, a unique role recognized by Sližienė was split into two different roles and mapped onto two different VerbNet equivalents. This had to be done with Sližienė’s role “quantitative” vs. VerbNet Asset and Attribute, and Sližienė’s “source”, mapped onto VerbNet “Material” and “Source”.

As the final step, the analysis is converted into JSON format. Both Sližienė’s roles and their VerbNet equivalents are presented in the output. Here is an example of the output for the sentence ‘The Labour Party urges the Social Democrats to look for a new Minister of Economy’:

---

3 The dependency tree has a distinct node for each word and punctuation sign in the sentence.
4 Nodes other than thematic-role bearing ones are labeled *pred (main predicate), *attr (attribute, genitive), and *time and *adv for adverbials.
5 For example, the Lithuanian equivalent of “do” can be daryti or padaryti: Sližienė’s dictionary does not include the latter as a separate entry; instead of this, the entry for daryti specifies additional piece of information that the perfective has the prefix pa-. Our table has daryti and padaryti as two separate entries; this results in a larger table but avoids the necessity for further morphological computation before the lookup.
6. Final Remarks

The preliminary evaluations carried out a few months ago with an earlier version suggest the precision to be more than 80% (about 75% for the $\theta$-role component), but the recall was only around 20%. Although the recall for the current version seems to be about 1/3, it has to be further improved. In the future, the results for each component will have to be described in detail, but the system is not sufficiently stable for this in-depth evaluation.

Beyond the actual webservice, the ultimate goal of this rule-based system is to automatically annotate syntactic corpora. Indeed, after the correction of the automatically generated data by some linguists, the corpus will give a basis for a future statistical analyser. Such a tool would improve the syntactic analysis in terms of accuracy and speed.

Acknowledgement

The engine belongs to the project Syntactic and Semantic Analysis System of the Lithuanian Language for Corpus, Internet, and Public Sector which is supported by the third priority (Information Society for All) of the Economic Growth Operational Programme.

References