Building a generative lexicon for Romanian

Anca Dinu
University of Bucharest
anca_d_dinu@yahoo.com

Abstract

We present in this paper an on-going research: the construction and annotation of a Romanian Generative Lexicon (RoGL). Our system follows the specifications of CLIPS project for Italian language. It contains a corpus, a type ontology, a graphical interface and a database from which we generate data in XML format.

1. Motivation.

We present in this article a part of an ongoing project of building a Romanian Generative Lexicon (RoGL), along the lines of Pustejovsky (2006).

Currently, there are a number of 'static' machine readable dictionaries for Romanian, such as Romanian Lexical Data Bases of Inflected and Syllabic Forms (Barbu, 2008), G.E.R.L. (Gavrila & Vertan, 2005), MULTEXT, etc. Such static approaches of lexical meaning are faced with two problems when assuming a fixed number of "bounded" word senses for lexical items:

- In the case of automated sense selection, the search process becomes computationally undesirable, particularly when it has to account for longer phrases made up of individually ambiguous words.

- The assumption that an exhaustive listing can be assigned to the different uses of a word lacks the explanatory power necessary for making generalizations and/or predictions about words used in a novel way.

Generative Lexicon (Pustejovsky, 1995) is a type theory with richer selectional mechanisms (see for instance Proceedings of The first/second/third International Workshop on Generative Approaches to the Lexicon 2001/2003/2005), which overcomes these drawbacks. The structure of lexical items in language over the past ten years has focused on the development of type structures and typed feature structures (Levin and Rappaport, 2005; Jackendoff, 2002). Generative Lexicon adds to this general pattern the notion of predicate decomposition. Lexicons built according to this approach contain a considerable amount of information and provide a lexical representation covering all aspects of meaning. In a generative lexicon, a word sense is described according to four different levels of semantic representation that capture the componential aspect of its meaning, define the type of event it denotes, describe its semantic context and positions it with respect to other lexical meanings within the lexicon.

GLs had been already constructed for a number of natural languages. Brandeis Semantic Ontology (BSO) is a large generative lexicon ontology and lexical database for English. PAROLE – SIMPLE – CLIPS lexicon is a large Italian generative lexicon with phonological, syntactic and semantic layers. The specification of the type system used both in BSO and in CLIPS largely follows that proposed by the SIMPLE specification (Busa et al., 2001), which was adopted by the EU-sponsored SIMPLE project (Lenci et al., 2000). Also, (Ruimy et al., 2005) proposed a method for semi-automated construction of a generative lexicon for French from Italian CLIPS, using a bilingual dictionary and exploiting the French-Italian language similarity.

Lexical resources, especially semantically annotated are notoriously effort and time consuming; thus, we tried to use as much already done work as possible in our effort to build a Romanian Generative Lexicon.

The rest of this paper is structured as it follows: in section 2, Generative Lexicon Theory is briefly outlined. Section 3 presents the motivation for choosing the CLIPS semantic structure for RoGL. The architecture of RoGL and the general methodology of construction and annotation is presented in section 4. In section 5 we discuss further work to be done.

2. Theoretical prerequisites: Generative Lexicon Theory

A predicative expression (such as a verb) has both an argument list and a body. Consider four possible strategies for reconfiguring the arguments-body structure of a predicate:

1. Atomic decomposition (do nothing – the predicate selects only the syntactic arguments):

$P(x_1,\ldots,x_n)$

2. Parametric decomposition (add arguments):

$P(x_1,\ldots,x_n) \rightarrow P(x_1,\ldots,x_n, x_{n+1},\ldots x_m)$

3. Predicative decomposition (split the predicate into subpredicates):

$P(x_1,\ldots,x_n) \rightarrow P_1(x_1,\ldots,x_n), P_2(x_1,\ldots,x_n), \ldots$

4. Full predicative decomposition (add arguments and split the predicate):

$P(x_1,\ldots,x_n) \rightarrow P_1(x_1,\ldots,x_n, x_{n+1},\ldots x_m), P_2(x_1,\ldots,x_n, x_{n+1},\ldots x_m),\ldots$

The theory uses the full predicative decomposition, with an elegant way of transforming the subpredicates into richer argument typing: Argument Typing as Abstracting from the Predicate:

$$\lambda x_2 \lambda x_1 [\Phi_1, \ldots \overbrace{\Phi_{x_1}}^{\tau}, \ldots \overbrace{\Phi_{x_2}}^{\sigma}, \ldots, \Phi_k] \Rightarrow$$

$$\lambda x_2 : \sigma\ \lambda x_1 : \tau [\Phi_1, \ldots, \Phi_k - \{\Phi_{x_1}, \Phi_{x_2}\}]$$

For example, possible types for the verb *sleep* are:

Approach	Type	Expression
Atomic	e -> t	λx[sleep(x)]
Predicatice	e -> t	λx[animate (x) ^ sleep(x)]
Enriched typing	anim -> t	λx : anim [sleep(x)]

Under such an interpretation, the expression makes reference to a type lattice of expanded types (Copestake and Briscoe, 1992;Pustejovsky and Boguraev, 1993).
Thus, generative Lexicon Theory employs the “Fail Early” Strategy of Selection, where argument typing can be viewed as pretest for performing the action in the predicate. If the argument condition (i.e., its type) is not satisfied, the predicate either: fails to be interpreted, or coerces its argument according to a given set of strategies. Composition is taken care of by means of typing and selection mechanisms (compositional rules applied to typed arguments).
Lexical Data Structures in GL:

1. Lexical typing structure: giving an explicit type for a word positioned within a type system for the language;
2. Argument structure: specifying the number and nature of the arguments to a predicate;
3. Event structure: defining the event type of the expression and any subeventual structure;
4. Qualia structure: a structural differentiation of the predicative force for a lexical item.

Argument and Body in GL:

$$\overbrace{\overbrace{\lambda x_n \dots \lambda x_1}^{AS}\,\overbrace{\lambda e_m \dots \lambda e_1}^{ES}}^{Environ}\qquad \overbrace{[Q_1 \wedge Q_2 \wedge Q_3 \wedge Q_4;\ C]}^{Body}$$

where AS: Argument Structure, ES: Event Structure, Qi: Qualia Structure, C: Constraints.
Qualia Structure:

1. Formal: the basic category which distinguishes it within a larger domain;
2. Constitutive: the relation between an object and its constituent parts;
3. Telic: its purpose and function, if any;
4. Agentive: factors involved in its origin or “bringing it about”.

A prototypical lexical entry for GL is given in fig. 1.
The Type Composition Language of GL:

1. e is the type of entities; t is the type of truth values. (σ and τ, range over simple types and subtypes from the ontology of e.)
2. If σ and τ are types, then so is σ -> τ ;
3. If σ and τ are types, then so is σ • τ ;
4. If σ and τ are types, then so is σ ΘQ τ, for Q = const(C), telic(T), or agentive(A).

Compositional Rules:

1. Type Selection: Exact match of the type.
2. Type Accommodation: The type is inherited.
3. Type Coercion: Type selected must be satisfied.

The domain of individuals (type e) is separated into three distinct type levels:

$$\left[\begin{array}{ll} \alpha & \\ \text{ARGSTR} = & \left[\begin{array}{l} \text{ARG1} = x \\ \dots \end{array}\right] \\ \text{EVENTSTR} = & \left[\begin{array}{l} \text{EVENT1} = e1 \\ \text{EVENT2} = e2 \end{array}\right] \\ \text{QUALIA} = & \left[\begin{array}{l} \text{CONST} = \textbf{what } x \textbf{ is made of} \\ \text{FORMAL} = \textbf{what } x \textbf{ is} \\ \text{TELIC} = e_2\textbf{: function of } x \\ \text{AGENTIVE} = e_1\textbf{: how } x \textbf{ came into being} \end{array}\right] \end{array}\right]$$

Figure 1. Prototipical lexical entry in GL

1. Natural Types: atomic concepts of formal, constitutive and agentive;
2. Artifactual Types: Adds concepts of telic;
3. Complex Types: Cartesian types formed from both Natural and Artifactual types.

3. Why choosing CLIPS architecture for RoGL

Creating a generative lexicon from scratch for any language is a challenging task, due to complex semantic information structure, multidimensional type ontology, time consuming annotation etc. Thus, in our effort to build a Romanian Generative Lexicon along the above theoretic lines, we made use of previous work both on Romanian static lexicon, such as (Barbu, 2008) and on existing generative lexicons for other languages such as Italian CLIPS or English BSO.
Our system follows closely the specifications of CLIPS project for Italian language. The reason for doing so is that we envision the possibility to semi-automatically populate RoGL using the massive Italian generative lexicon CLIPS and a quality bilingual dictionary.
The idea is not original: such a research exists for French, exploiting the French-Italian language similarity, with encouraging results (Ruimy et al, 2005). The authors proposed a method based on two complementary strategies (cognate suffixes and sense indicators) for relating French word senses to the corresponding CLIPS semantic units. The cognate strategy proposed is guided by the following two hypotheses:

- morphologically constructed words usually have sense(s) that are largely predictable from their structure;
- Italian suffixed items have one (or more) equivalent(s) – constructed with the corresponding French suffix – that cover(s) all the senses of the Italian word.

If an Italian CLIPS word has, in the bilingual dictionary, the same translation for all its senses, this unique French equivalent will share with the Italian word all the SIMPLE-CLIPS semantic entries.

We may employ the same strategy to obtain Romanian semantically annotated units from their Italian counterpart. The fact that Romanian is in the same group of romance languages creates the morpho-syntactic premises to obtain similar results.

The cognates approach is rather easy to implement (and yields expected higher recall then sense indicator method), based, for example, on cognateness of suffixes from Romanian and Italian (such as –ie, -zione; -te, -tà). For the other words and for those constructed words that have more than one translation, the cognate method results inadequate and the sense indicator method takes over. The sense indicator method is more demanding, but has a higher precision. A specific algorithm for Romanian - Italian needs to be design and implemented.

4. Architecture and Implementation

Our system contains a corpus, an ontology of semantic types, a graphical interface and a database from which we generate data in XML format (figure 2).

We used the RORIC-LING Romanian corpus (Hristea & Popescu, 2003) to feed the annotation graphical interface with lexical items in their context (phrase they appear in). The corpus is rather small (98 newspaper texts), but it has the advantage that is already syntactically annotated in XML. We proceed with the annotation of lexical units in their frequency order.

The type ontology we choose is very similar with the CLIPS ontology. It has a top node, with types Telic, Agentive, Constitutive and Entity, as daughters. The types Telic, Agentive and Constitutive are intended to be assigned as types only for lexical units that can be exclusively characterized by one of them. Type Entity has as subtypes Concrete_entity, Abstract_entity, Property, Representation, and Event. In all, the ontology has 144 types and can be further refined in a subsequent phase of RoGL, if the annotation process supplies evidences for such a necessity.

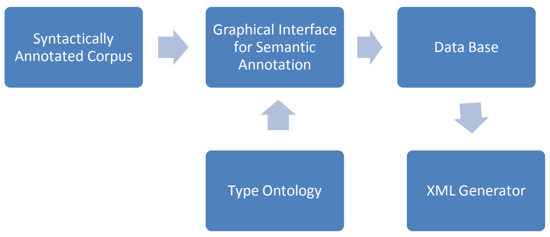

Syntactically Annotated Corpus
Graphical Interface for Semantic Annotation
Data Base
Type Ontology
XML Generator

Figure 2. Architecture of RoGL.

The first task the annotator has to deal with is to choose one of the meanings of the lexical unit. The annotator sees a phrase with the target word highlighted. To help the annotator, a gloss comprising the possible meanings from an electronic dictionary pops up. Here we are interested in regular polysemy (such as bank: institution or chair), not the different meaning levels of the same lexeme (such as book: the physical object or the information), aspect which is to be described later by specifying the semantic type of the lexical item as complex. We will record in the data base different entries for different senses of a polysemantic lexical entry.

The semantic type of the lexical unit is first chosen from a list of 17 types. Only if the annotator cannot find the right type to assign to the lexical unit, he may consult the complete ontology. Thus, the complexity of annotation task remains tractable: the annotator does not have to bother with the inheritance structure or with over 100 types to choose from. The 17 initial types are the ones in Brandeis Shallow Ontology (table 1), a shallow hierarchy of types selected for their prevalence in manually identified selection context patterns. They were slightly modified to mach our ontology and we expect to modify them again to fit our Romanian data, once we have our own annotations statistics. It is important to notice that the same lexical unit is presented several times to the annotator in a different context (phrase). For the same disambiguated meaning, the annotator may enhance the existing annotation, adding for example another type for the lexical unit (see the dot operator for complex types in chapter 2).

Top Types	**Abstract Entity Subtypes**
abstract entity	attitude
human	emotion
animate	property
organization	obligation
physical object	rule
artifact	
event	
proposition	
information	
sensation	
location	
time period	

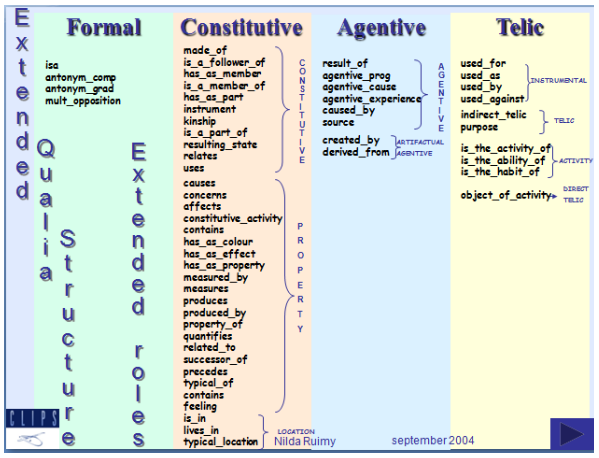
Table 1: Type System for Annotation

The part of speech is automatically taken from the corpus. The annotator has to refine it further into one of the following pos tags, which are not present in the corpus, such as: intransitive verb, transitive verb, ditranzitive verb, unpredicative noun, predicative noun and adjective. Depending on the particular pos selected for a lexical unit, it's predicative structure modifies. Accordingly, once one of the pos tags was selected, our graphical interface automatically creates a template matching argument structure with no arguments, with Arg0, with Arg0 and Arg1, or with Arg0, Arg1 and Arg2.

The event type is selected from a drop down list comprising process, state and activity.

The Qualia Structure in RoGL follows the CLIPS extended qualia structure (figure 3): each of the four qualia relations has a list of extended relations which the

annotator has to choose from. The choice may be obligatory, optional or multiple.
As to the Predicative Representation, it describes the semantic scenario the word sense considered is involved in and characterizes its participants in terms of thematic roles and semantic constraints.


Extended Qualia Structure Extended roles

Formal
isa
antonym_comp
antonym_grad
mult_opposition

Constitutive
made_of
is_a_follower_of
has_as_member
is_a_member_of
has_as_part
instrument
kinship
is_a_part_of
resulting_state
relates
uses
} CONSTITUTIVE
causes
concerns
affects
constitutive_activity
contains
has_as_colour
has_as_effect
has_as_property
measured_by
measures
produces
produced_by
property_of
quantifies
related_to
successor_of
precedes
typical_of
contains
feeling
} PROPERTY
is_in
lives_in
typical_location
} LOCATION

Agentive
result_of
agentive_prog
agentive_cause
agentive_experience
caused_by
source
} AGENTIVE
created_by } ARTIFACTUAL
derived_from } AGENTIVE

Telic
used_for
used_as
used_by
used_against
} INSTRUMENTAL
indirect_telic
purpose
} TELIC
is_the_activity_of
is_the_ability_of
is_the_habit_of
} ACTIVITY
object_of_activity DIRECT TELIC

CLIPS
Nilda Ruimy september 2004

Figure 3. Extended qualia relations from CLIPS

The annotator has to choose the lexical predicate the semantic unit relates to and the type of link between them (master, event, process or state nominalization, adjective nominalization, agent nominalization, patient nominalization, instrument nominalization, other nominalization). In the data base, we store the predicates separately from the semantic units.
For example, the predicate *a construi* (to build) is linked to USem *constructie* (construction - building) by a patient nominalization link, to USem *construire* (construction - process) by a process nominalization link, to USem *constructor* (constructor) by an agent nominalization link and to USem *construi* (to build) by a master link.

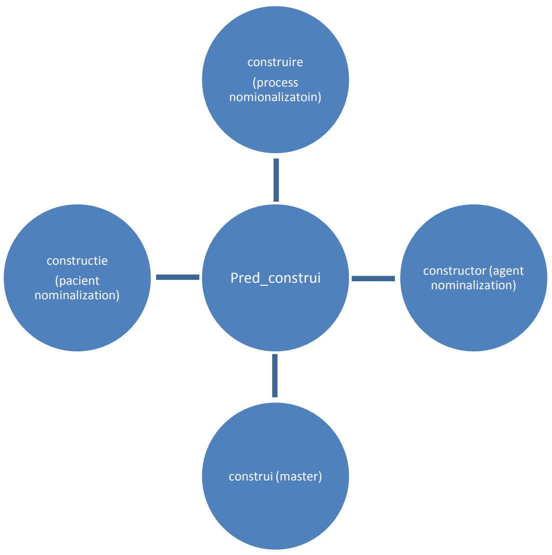
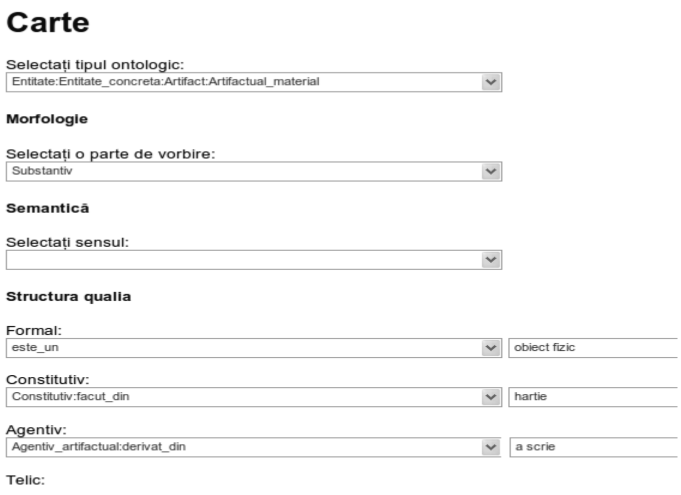

construire (process nomionalizatoin)
constructie (pacient nominalization)
Pred_construi
constructor (agent nominalization)
construi (master)

Figure 4. Semantic frame for the predicate *a construi.*

The argument structure annotation consists of choosing for each argument its type from the ontology (the semantic constraints of the semantic unit) and their thematic roles from the thematic roles list: Protoagent (arg0 of kill), Protopatient (arg1 of kill), SecondParticipant (arg2 of give), StateOfAffair (arg2 of ask), location (arg2 of put), Direction (arg2 of move), Origin (arg1 of move), Kinship (arg0 of father), HeadQuantified (arg0 of bottle).
Figure 4 depicts a fragment of the annotation process for a noun (carte - book):


Carte

Selectați tipul ontologic:
Entitate:Entitate_concreta:Artifact:Artifactual_material

Morfologie

Selectați o parte de vorbire:
Substantiv

Semantică

Selectați sensul:

Structura qualia

Formal:
este_un | obiect fizic

Constitutiv:
Constitutiv:facut_din | hartie

Agentiv:
Agentiv_artifactual:derivat_din | a scrie

Telic:

Figure 4. A fragment of annotation process.

To implement the generative structure and the composition rules, we chose a functional programming language of the Lisp family, namely Haskell. The choice of functional programming is not accidental. With Haskell, the step from formal definition to program is particularly easy. Most current work on computational semantics uses Prolog, a language based on predicate logic and designed for knowledge engineering. Unlike the logic programming paradigm, the functional programming paradigm allows for logical purity. Functional programming can yield implementations that are remarkably faithful to formal definitions. In fact, Haskell is so faithful to its origins that it is purely functional, i.e. functions in Haskell do not have any side effects. (However, there is a way to perform computations with side effects, like change of state, in a purely functional fashion).
Our choice was also determined by the fact that reducing expressions in lambda calculus (obviously needed in a GL implementation), evaluating a program (i.e. function) in Haskell, and composing the meaning of a natural language sentence are, in a way, all the same thing.
The Haskell homepage http://www.haskell.org was very useful. The definitive reference for the language is (Peyton Jones2003). Textbooks on functional programming in Haskell are (Bird, 1998) and (Hutton, 2007).

5. Further work

As we said, this is an ongoing project. Most importantly, we need to annotate more lexical entries. The manual annotation, although standardized and mediated by the graphical interface is notoriously time consuming especially for complex information such as those required by a generative lexicon. We plan automate the process to some extent, taking advantage of the existing work for Italian. Thus, the CLIPS large and complex generative lexicon may be used in an attempt to automatically populate a Romanian GL. A feasibility study is necessary to assess the potential coverage of such a method. However, the final annotation, we believe, is to be done manually.

6. Acknowledgements

This work was supported by CNCSIS –UEFISCSU, project number PNII – IDEI 228/2007.

7. References

Barbu, A. M. "Romanian Lexical Data Bases: Inflected and Syllabic Forms Dictionaries", LREC 2008, May, 28-30, Marrakech, Marocco, 2008.

Bird, R. Introduction to Functional Programming Using Haskell. Prentice Hall,1998.

Busa, F., Calzolari, N., Lenci, A.: Generative Lexicon and the SIMPLE Model; Developing Semantic Resources for NLP, in Bouillon P. and Busa F. (eds.), The Language of Word Meaning, Cambridge University Press, pp. 333-349, 2001.

Copestake, A. and T. Briscoe "Lexical Operations in a Unificationbased Framework," in J.Pustejovsky and S. Bergler, eds., Lexical Semantics and Knowledge Reperesentation, Springer Verlag, Berlin, 1992.

Hristea, F., M. Popescu A Dependency Grammar Approach to Syntactic Analysis with Special Reference to Romanian, in: Building Awareness in Language Technology (editori Florentina Hristea si Marius Popescu). Bucuresti, Editura Universitatii din Bucuresti, p. 9-34, 2003.

Hutton, G. Programming in Haskell. Cambridge University Press, 2007.

Jackendoff, R. Foundations of language: Brain, meaning, grammar, evolution. Oxford, UK: Oxford University Press, 2002.

Lenci A., Bel N., Busa F., Calzolari N., Gola E., Monachini M., Ogonowsky A., Peters I., Peters W., Ruimy N., Villegas M., Zampolli A. "SIMPLE: A General Framework for the Development of Multilingual Lexicons", International Journal of Lexicography, XIII (4): 249-263, 2000.

Levin, B. and M. Rappaport Hovav. Argument Realization, Cambridge University Press, Cambridge, UK., 2005

Peyton Jones S., editor. Haskell 98 Language and Libraries. Cambridge, University Press, 2003.

Pustejovsky, J. The Generative Lexicon, Cambridge, MA: MIT Press, 1995.

Pustejovsky, J. 2007. Type Theory and Lexical Decomposition. In P. Bouillon and C. Lee, editors, Trends in Generative Lexicon Theory. Kluwer Publishers, 2007.

Pustejovsky, J., A. Rumshisky, J. Moszkowicz, and O. Batiukova. GLML: Annotating argument selection and coercion. IWCS-8: Eighth International Conference on Computational Semantics, 2009.

Ruimy,N., P. Bouillon and B. Cartoni. Inferring a semantically annotated generative French lexicon from an Italian lexical resource. In: Third International Workshop on Generative Approaches to the Lexicon : May 19-21, 2005, Geneva, Switzerland. 218-226, 2005.

Semi-automatic Derivation of a French Lexicon from CLIPS, Ruimy N., Bouillon P. and Cartoni B., in: Proceedings of LREC04, Lisbon, 2004.

Vertan, C. von Hahn. and Monica Gavrila. Designing a parole/simple german-english-romanian lexicon. In Language and Speech Infrastructure for Information Access in the Balkan Countries Workshop Proceedings - RANLP 2005, Borovets, Bulgaria, September 2005.