ANNOTATING A ROMANIAN LEXICON IN A GENERATIVE FRAMEWORK

Anca Dinu

Abstract: We present in this paper an on-going research: the construction and annotation of a Romanian Generative Lexicon (RoGL), following the generative lexicon theory. Our system follows the specifications of CLIPS project for Italian language. It contains a corpus, an ontology of semantic types, a graphical interface and a database from which we generate data in XML format. We describe here the graphical interface structure as well as functionality and the annotation procedure.

Keywords: generative lexicon, semantic annotation, graphical interface, Romanian language

1. Introduction

We present in this paper[1] an on-going research: the construction and annotation of a Romanian Generative Lexicon (RoGL).

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. (Vertan et al. 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.

The Generative Lexicon (Pustejovsky 1995) is a type theory with richer selectional mechanisms, 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 Hovav 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 the 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,

[1] This work was supported by CNCSIS-UEFISCSU, project PNII-IDEI 228/2007.

(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 available work as much as possible in our effort to construct and annotate 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 describes our general methodology and architecture for RoGL construction and annotation. Section 4 describes the graphical interface and the annotation tasks. Finally, in section 5, we discuss further work to be done.

2. Generative lexicon: Overview

A predicative expression (such as a verb) has both an argument list and a body:

(1)

$$\overbrace{\lambda x_i}^{Args}\ \overbrace{[\Phi]}^{Body}$$

Consider four possible strategies for reconfiguring the args-body structure of a predicate:

(i) atomic decomposition (do nothing – the predicate selects only the syntactic arguments): $P(x_1,\ldots,x_n)$

(ii) parametric decomposition (add arguments): $P(x_1,\ldots,x_n)$ -> $P(x_1,\ldots,x_n, x_{n+1},\ldots x_m)$

(iii) predicative decomposition (split the predicate into subpredicates): $P(x_1,\ldots,x_n)$ -> $P_1(x_1,\ldots,x_n), P_2(x_1,\ldots,x_n),\ldots$

(iv) full predicative decomposition (add arguments and split the predicate): $P(x_1, ,\ldots,x_n)$ -> $P_1(x_1,\ldots,x_n, x_{n+1},\ldots x_m),P_2(x_1,\ldots,x_n, x_{n+1},\ldots x_m),\ldots$

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:

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

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

Table 1

Approach	Type	Expression
Atomic	$e \rightarrow t$	λ*x*[*sleep*]
Predicative	$e \rightarrow t$	λ*x*[*animate*(*x*) ∧ *sleep*(*x*)]
Enriched typing	*anim* → *t*	λ*x*: *anim*[*sleep*(*x*)]

Under such an interpretation, the expression makes reference to a type lattice of expanded types (cf. Copestake and Briscoe 1992).

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).

The lexical data structures in GL are composed of: (i) lexical typing structure, giving an explicit type for a word positioned within a type system for the language; (ii) argument structure, specifying the number and nature of the arguments to a predicate; (iii) event structure, defining the event type of the expression and any subeventual structure; (iv) qualia structure: a structural differentiation of the predicative force for a lexical item.

Schematically, the argument and body in GL look like this:

(2)

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

where AS = argument structure, ES = event structure, Qi = qualia structure and C = Constraints. The original part of the GL structure is Qualia Structure, composed of: (i) Formal: the basic category which distinguishes it within a larger domain; (ii) Constitutive: the relation between an object and its constituent parts; (iii) Telic: its purpose and function, if any; (iv) Agentive: factors involved in its origin or "bringing it about". A prototypical lexical entry for GL is given below:

Figure 1: Prototypical lexical entry

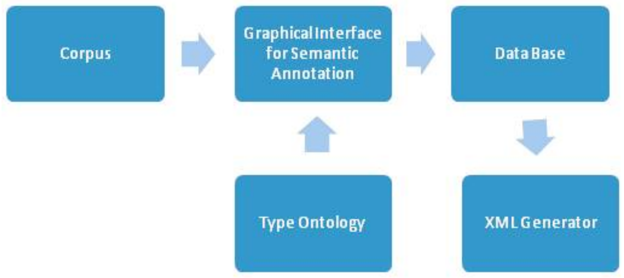
$$\begin{bmatrix} \alpha & \\ \text{ARGSTR} = & \begin{bmatrix} \text{ARG1} = x \\ \ldots \end{bmatrix} \\ \text{EVENTSTR} = & \begin{bmatrix} \text{EVENT1} = e1 \\ \text{EVENT2} = e2 \end{bmatrix} \\ \text{QUALIA} = & \begin{bmatrix} \text{CONST} = \textbf{what } x \textbf{ is made of} \\ \text{FORMAL} = \textbf{what } x \textbf{ is} \\ \text{TELIC} = e_2\textbf{: function of } x \\ \text{AGENTIVE} = e_1\textbf{: how } x \textbf{ came into being} \end{bmatrix} \end{bmatrix}$$

The Type Composition Language of GL is: (i) e is the type of entities; (ii) t is the type of truth values (σ and τ, range over simple types and subtypes from the ontology of e.); (iii) If σ and τ are types, then so is $\sigma \rightarrow \tau$; (iv) If σ and τ are types, then so is $\sigma \bullet \tau$; (v) If σ and τ are types, then so is $\sigma \odot_Q \tau$, for Q = const(C), telic(T), or agentive(A). Finally, the Compositional Rules of the theory are: (i) type selection: exact match of the type; (ii) type accommodation: the type is inherited; (iii) type coercion: the type selected must be satisfied. The domain of individuals (type e) is separated into three distinct type levels: (i) natural types: atomic concepts of formal, constitutive and agentive; (ii) artefactual types: concepts of telic; (iii) complex types: Cartesian types formed from both natural and artefactual types.

3. Methodology and architecture of RoGL

Creating a generative lexicon for any language is a challenging task, due to complex semantic information structure, multidimensional type ontology, time consuming annotation etc. Our system follows the specifications of CLIPS project for Italian language. It contains a corpus, an ontology of semantic types, a graphical interface and a database from which we generate data in XML format (figure 2):

Figure 2: Architecture of RoGL



As a starting point for the annotation process, we used the RORIC-LING Romanian corpus (Hristea and 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: we chose to first annotate the first frequent 100 verbs, 100 nouns and 20 adjectives from the corpus.

The annotation is to be done web-based, via a graphical interface, to avoid compatibility problems. The interface and the data base where the annotated lexical entries will be stored and processed are hosted on the server of Faculty of Mathematics and Informatics, University of Bucharest[2]. Each annotator receives a username and a password from the project coordinator in order to protect already introduced data and also to protect against introducing erroneous data.

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.

To implement the generative structure and the composition rules, we chose a functional programming language of the Lisp family, namely Haskell[3]. 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. 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.

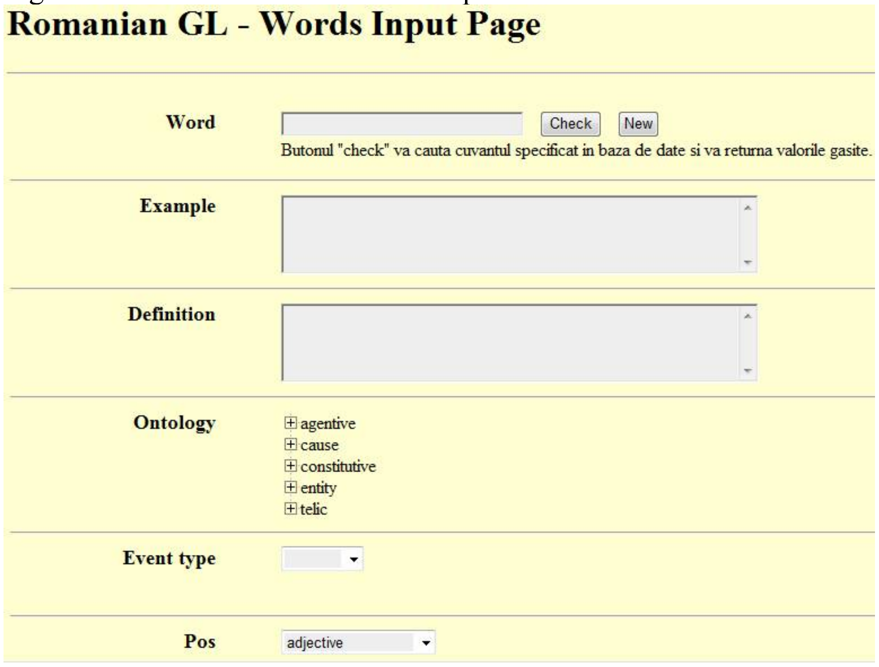
[2] At http://ro-gl.fmi.unibuc.ro.

[3] The Haskell homepage http://www.haskell.org was very useful. The definitive reference for the language is Peyton Jones (2003). Textbooks on functional programming in Haskell are Bird (1998) and Hutton (2007).

4. The annotation

We describe here the graphical interface structure and functionality as well as the annotation procedure. The first task the annotator has to deal with is to choose one of the meanings of the possibly homonym lexical unit. The annotator sees a phrase with the target word highlighted. To help the annotator, a gloss comprising the possible different meanings from an electronic dictionary pops up. Here we are interested only in distinguishing between different meanings of homonym words (same orthography and pronunciation, completely different meaning, such as bank: institution or chair), not the different meaning levels of the same lexeme (such as book: the physical object or the information). The former aspect of meaning is to be described by specifying the type of the lexical item as complex, i.e. composed by two or more different semantic types from the ontology.

Figure 3: First tasks of the annotation process



Romanian GL - Words Input Page

Word	[ ] Check New Butonul "check" va cauta cuvantul specificat in baza de date si va returna valorile gasite.
Example	
Definition	
Ontology	⊞ agentive ⊞ cause ⊞ constitutive ⊞ entity ⊞ telic
Event type	
Pos	adjective

The next step for the annotator is to choose the type of the lexical unit from a tree structure of 144 types that compose the type ontology (figure 3). As the annotation process progresses, we will be able to propose to the annotator a short list of types to choose from, based on a statistics of most frequent types selected until the moment of annotation. So, only if the annotator cannot find the right type to assign to the lexical unit in the proposed short list, he has access to the whole type ontology. Thus, the complexity of annotation task remains tractable: the annotator does not have to bother with the inheritance structure or with too many types to choose from. For example, in Brandeis Shallow Ontology (BSO), a shallow hierarchy of 17 types was set (table 1). These types were selected for their prevalence in manually identified selection context patterns. 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 section 2). The classical example is the semantic type received in generative lexicon theory for the lexical unit *book*: *physical object @ information*, which is a complex type, obtained from two basic (natural or artefactual) types and the dot operator @.

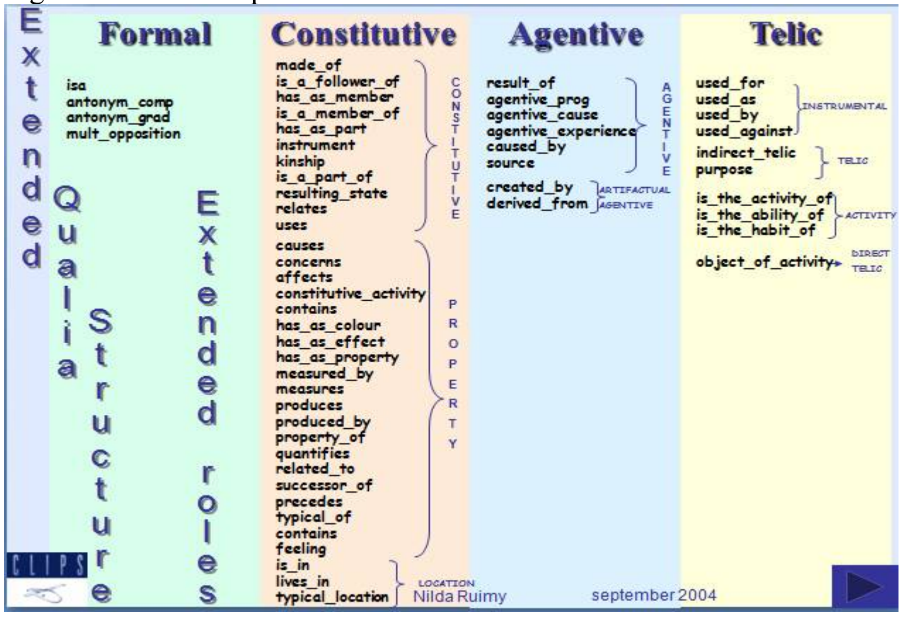
Table 2: Type system for annotation in BSO

Top types	**Abstract entity subtypes**
abstract entity	attitude
human	emotion
animate	property
organization	obligation
physical object	rule
artefact	
event	
proposition	
information	
sensation	
location	
time period	

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 as such: intransitive verb, transitive verb, ditranzitive verb, non-redicative noun, predicative noun (such as deverbals; collective simple nouns, e.g. *grup* 'group'; nouns denoting a relation, e.g. *mamă* 'mother'; quantity, e.g. *sticlă* 'bottle'; part, e.g. *bucată* 'piece'; unit of measurement, e.g. *metru* 'metre'; property, e.g. *frumusețe* 'beauty'), adjective. Depending on the particular pos selected for a lexical unit, its 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 annotator is then asked to specify the qualia structure of the current word. The Qualia Structure in RoGL follows the CLIPS extended qualia structure (figure 4): each of the four qualia roles has a (dropdown) list of extended roles which the annotator has to choose from. The choice may be obligatory, optional or multiple.

Figure 4: Extended qualia roles from CLIPS



Extended Qualia Structure Extended roles

Formal
isa
antonym_comp
antonym_grad
mult_opposition

Constitutive
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
PROPERTY: 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
LOCATION: is_in, lives_in, typical_location

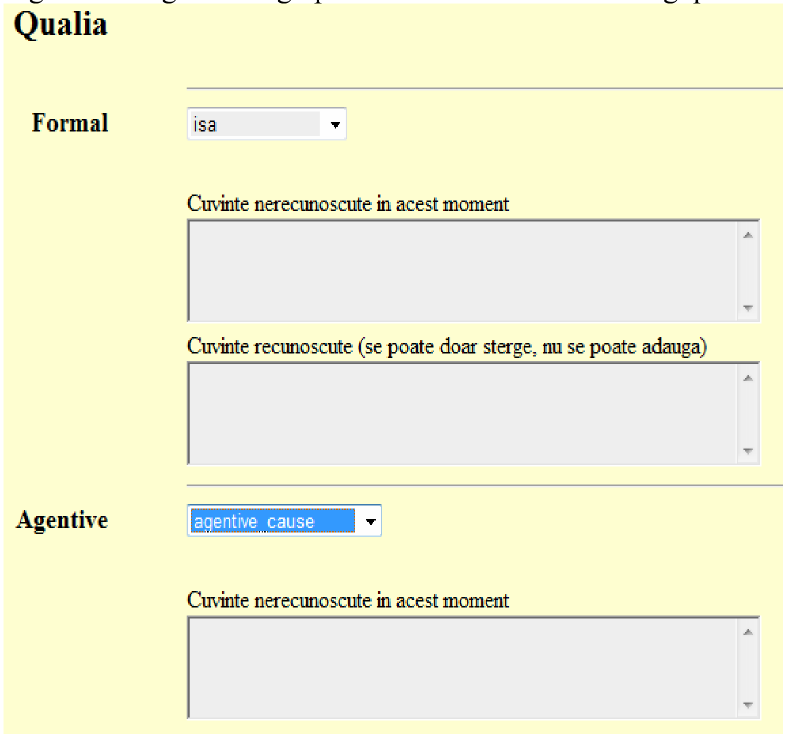
Agentive
AGENTIVE: result_of, agentive_prog, agentive_cause, agentive_experience, caused_by, source
ARTIFACTUAL AGENTIVE: created_by, derived_from

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

CLIPS
Nilda Ruimy september 2004

Then the annotator has to provide the words which are in the specified relation with the current word. Here a distinction is to be made between existing words (already introduced in the data base) and words not jet introduced. For existing words, a link between each of them and the current word is automatically created. For the others, a procedure of verification for the data base has to be run at some time intervals, in order to check and update the existing links, so that words in the lexicon become maximally connected. Figure 5 depicts a fragment of the graphical interface for annotating the qualia structure:

Figure 5: Fragment of graphical interface for annotating qualia structure



Qualia

Formal: isa

Cuvinte nerecunoscute in acest moment

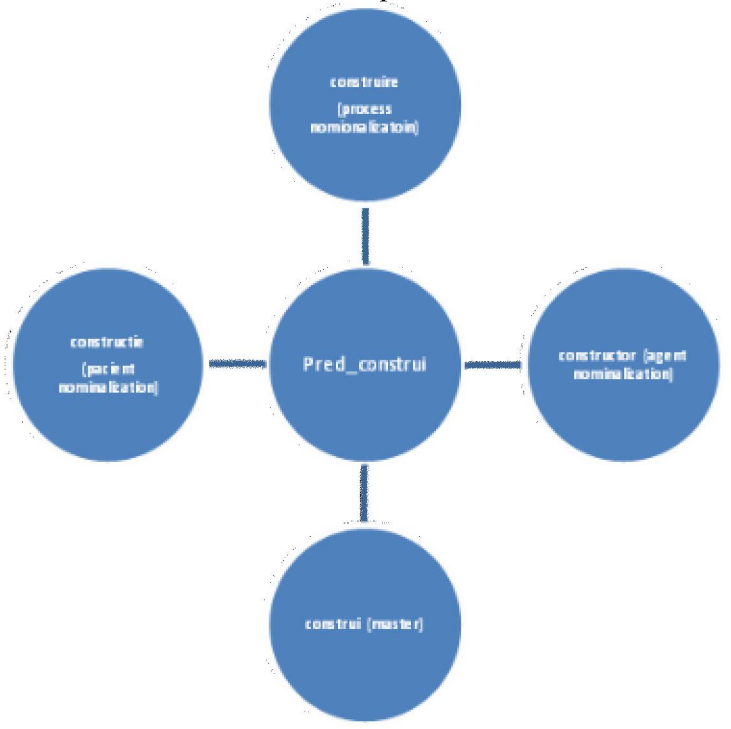
Cuvinte recunoscute (se poate doar sterge, nu se poate adauga)

Agentive: agentive_cause

Cuvinte nerecunoscute in acest moment

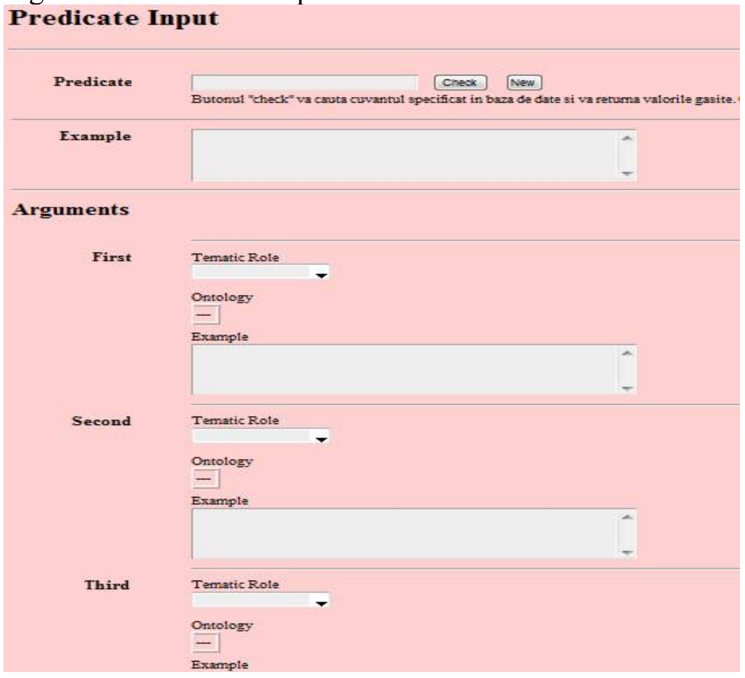
The Predicative Representation describes the semantic scenario the considered word sense is involved in and characterizes its participants in terms of thematic roles and semantic constraints. We make use again of the expertise of the CLIPS developers in adopting an adequate predicative representation for RoGL. In SIMPLE project, the predecessor of CLIPS project, only the predicative lexical units (units that subcategorize syntactic arguments) receive a predicative representation: for example, a word like *constructor* (which is not the head of a syntactic phrase) is not linked with the predicate *to construct*. In CLIPS (and also in RoGL), the non-predicative lexical units may be linked (when the annotator decides) to a predicative lexical unit, thus constructor is linked by an AgentNominalization type of link to the predicative lexical unit to construct, so it fills the arg0 of this predicate. The type of link Master is to be chosen between a predicative unit and its predicative structure (representation). Thus, in the ideal case, a semantic frame such as to construct (the predicate), construction (pacient or process nominalization) and constructor (agent nominalization) will end up being connected (with the proper semantic type of link) in the data base.

Figure 6: The semantic links for the predicate *a construi* ‘to construct’



The annotation task for the predicative structure consists of choosing for each argument (one, two or three) the semantic type from the ontology list 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 7: Annotation of predicative structure


Predicate Input

Predicate

Check New

Butonul "check" va cauta cuvantul specificat in baza de date si va returna valorile gasite.

Example

Arguments

First

Tematic Role

Ontology

Example

Second

Tematic Role

Ontology

Example

Third

Tematic Role

Ontology

Example

5. Conclusions

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 to use machine learning techniques to automate the process, 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. 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 fact that Romanian is in the same group of Romance languages creates the morpho-syntactic premises to obtain similar results. However, the final annotation, we believe, is to be done manually.

Anca Dinu
University of Bucharest
Faculty of Foreign Languages and Literatures
anca_d_dinu@yahoo.com

References

Barbu, A. M. 2008. Romanian lexical data bases: Inflected and syllabic forms dictionaries. <http://www.lrec-conf.org/proceedings/lrec2008>.

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

Busa, F., Calzolari, N., Lenci, A. 2001. Generative Lexicon and the SIMPLE Model: Developing semantic resources for NLP. In P. Bouillon and F. Busa (eds.), *The Language of Word Meaning*, 333-349. Cambridge: Cambridge University Press.

Copestake, A. and Briscoe, T. 1992. Lexical operations in a unification-based framework. In J. Pustejovsky and S. Bergler (eds.), *Lexical Semantics and Knowledge Representation*, 101-119. Berlin: Springer Verlag.

Hristea, F., Popescu, M. 2003. A Dependency Grammar approach to syntactic analysis with special reference to Romanian. In F. Hristea and M. Popescu (eds.), *Building Awareness in Language Technology*, 9-34. Bucharest: Editura Universitatii din Bucuresti.

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

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

Lenci, A., Bel, N., Busa, F., Calzolari, N., Gola, E., Monachini, M., Ogonowsky, A., Peters, I., Peters, W., Ruimy, N., Villegas, M., Zampolli, A. 2000. SIMPLE: A general framework for the development of multilingual lexicons. *International Journal of Lexicography* XIII (4): 249-263.

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

Peyton Jones S., (ed.). 2003. *Haskell 98 Language and Libraries*. Cambridge: Cambridge University Press.

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

Ruimy, N., Bouillon, P. and Cartoni, B. 2005. 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*, 218-226.

Vertan, C., von Hahn, W. and Gavrilă, M. 2005. Designing a parole/simple German-English-Romanian lexicon. In *Language and Speech Infrastructure for Information Access in the Balkan Countries Workshop Proceedings – RANLP 2005, September 2005, Borovets*, *Bulgaria*, 82-87.