ODL: An Object Description Language for Lexical Information

Michael Rosner

Department of Artificial Intelligence
University of Malta
Msida MSD 2080, Malta
mike.rosner@um.edu.mt

Abstract

This paper describes ODL, a description language for lexical information that is being developed within the context of a national project called MLRS (Maltese Language Resource Server) whose goal is to create a national corpus and computational lexicon for the Maltese language. The main aim of ODL is to make the task of the lexicographer easier by allowing lexical specifications to be set out formally so that actual entries will conform to them. The paper describes some of the background motivation, the ODL language itself, and concludes with a short example of how lexical values expressed in ODL can be mapped to an existing tagset together with some speculations about future work.

1. Introduction

This paper introduces ODL 1.0, a description language for lexical information that has been developed for MaltiLex, the computational lexicon originally conceived in 1998 (Rosner et al., 1998) and now being further developed within the scope of MLRS (Rosner et al., November 2006), a national project aimed at developing a server for Maltese language resources.

2. Goal of the paper

The main goal of this paper is to present the ODL language as a viable description language for lexical information. This is done in section 5.. Subsidiary goals are (i) to situate the work within the general context of research in the area of computational lexicography (section 3. (ii) to show how the system can be used for the description of Maltese lexical values (section 6.) and (iii) to illustrate the connection between that system of lexical values and the tagset that is being developed for Maltese. This last goal, described in section 7. is work in progress.

3. Computational Lexicography

Computational lexicography has a long history which is almost as old as electronic computers. Previous research most relevant to the task of creating machine-tractable lexicons falls into three broad categories:

- **Work on extracting entries from machine-readable dictionaries**. Much of the early work was structured around different aspects of machine readable dictionaries including research on the taxonomic structure of existing dictionaries by Amsler (1980), the syntactic structure of entries by newciteMichiels:1982 and on algorithms for the automatic extraction of information from that dictionary (Boguraev et al., 1987). A development of this approach, directed towards the automated construction of dictionary entries for a machine translation system using LDOCE as a source, is described by Farwell et al. (1992).
- **Work on word-senses and semantic relationships between them**. The key system here is the WordNet project (Fellbaum, 1998) and dervatives such as Eurowordnet (Vossen, 1997). Any WordNet is less a dictionary of words than a database system for handling the definition of word senses *synsets* and the relationships between them.
- **Work on lexicon editors and support for lexicographers**. Lexicography support tools include early systems like Shoebox, still available from the Summer Institute of Linguisitics[1] that help field linguists and anthropologists integrate various lexical, cultural, grammatical information, to more more modern tools which help the lexicographer analyse word behaviour within corpora. A good example of such a tool is the Sketch Engine (Kilgarriff et al., 2004) which provides one-page summaries of a word's grammatical and collocational behaviour.

  Tsalidis and his colleagues (Tsalidis et al., 2004) have been working on a large-scale electronic lexicon for Greek and have developed `LexEdit`, a lightweight lexicon editor which was used for the initial definition of the lexicon entries.

  Finally LEXUS (Marc et al., 2006) is a recent addition to the set of lexicographic tools being developed at the Max-Planck Institute. It is regarded by its authors as an implementation of the LMF, the lexical markup framework model that is being developed under the ISO TC37SC4[2] and in many ways comes closest to the work described in this paper in so far as it provides a web-based tool for developing lexical entries under the control of a formal specification.

4. Maltilex

MaltiLex is a computational lexicon for the Maltese language with the following charactistics:

- **Full form:** Maltese is a language with a highly complex and rich morphology. There is some disagreement about the best way to handle morphology from a

[1] `http://www.sil.org/computing/shoebox/`
[2] `www.tc37sc4.org`

computational perspective. We have therefore decided to sidestep the issue by first defining a full-form core lexicon.

- **Wide coverage:** being part of a national project, MaltiLex is intended to cover a reasonably large subset of the language. In practical terms, we are allowing for the possibility of several million entries, and hence, efficiency of representation is an important consideration.

- **Expert content provision:** although large numbers of potential entries can be harvested from the internet and other sources by automatic means, we believe that the quality of content still depends largely on input from those having expert linguistic knowledge. We therefore wish to design an interface that facilitates interaction with such experts.

- **Efficient lookup:** Amongst the tasks carried out by linguists is search for words according to complex boolean criteria. We wish to make sure that this kind of search is performed as efficiently as possible.

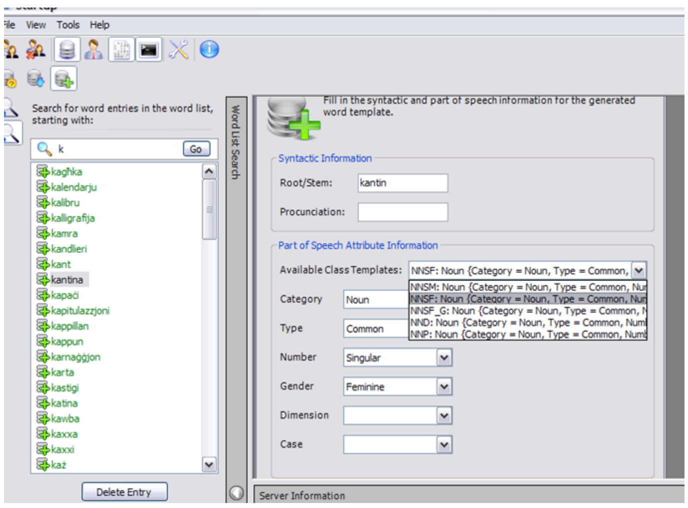
4.1. Implementating MaltiLex

The heart of MaltiLex is actually an SQL database which implements a relation between *lexical forms*, which is simply a string that might be a word form or part of a word, and *lexical information* in the form of a set of attribute-value pairs. For example, the Maltese word *klieb* (books) is a plural noun. This could be expressed by associating the *attributes* `cat` and `num` with the *values* `N` and `plur` respectively.

More formally, we can say that a lexicon is an N-tuple $< L, A, V, R >$ where L is a set of lexical forms, A is a set of attributes, V is a set of atomic values, $R \subseteq L \times F$ is a relation which pairs lexical forms to function $F \subseteq A \Rightarrow V$. Hence the lexical information associated with *klieb* can be regarded as the *function* {(`num`,`plur`),(`cat`,`N`)} which maps the attributes `cat`, `num` to the values `plur`, and `N` respectively. This view of the lexicon has a long tradition as reflected in FUG (Kay, 1984), LFG (Kaplan and Bresnan, 1982) HPSG (Pollard and Sag, 1987) and more recent developments.

Given our desire to support wide coverage of the language and efficient search, MaltiLex has been implemented using a relational database in which lexical information is represented using a fixed length bit-field. This makes it very easy for a linguist to specify boolean search criteria and for queries about words to be efficiently executed. However, it does lead to certain built-in limitations.

One is that the number of attribute-value pairs cannot exceed a certain fixed value. This has not turned out to be a problem so far given the special tagset for Maltese that we have developed (Gatt et al., 2003). Another, potentially more serious problem is that the values themselves are atomic, so that the system cannot support recursive feature structures as used within frameworks such as HPSG (Pollard and Sag, 1987), LFG (Kaplan and Bresnan, 1982) and TFS (Emele, 1994).



Search for word entries in the word list, starting with: k

kagħka
kalendarju
kalibru
kalligrafija
kamra
kandlieri
kant
kantina
kapaċi
kapitulazzjoni
kappillan
kappun
karnaġġjon
karta
kastigi
katina
kawba
kaxxa
kaxxi
każ

Fill in the syntactic and part of speech information for the generated word template.

Syntactic Information

Root/Stem: kantin
Procunciation:

Part of Speech Attribute Information

Available Class Templates: NNSF: Noun {Category = Noun, Type = Common,
NNSM: Noun {Category = Noun, Type = Common, Nu
NNSF: Noun {Category = Noun, Type = Common, Nu
NNSF_G: Noun {Category = Noun, Type = Common, N
NND: Noun {Category = Noun, Type = Common, Num
NNP: Noun {Category = Noun, Type = Common, Num

Category: Noun
Type: Common
Number: Singular
Gender: Feminine
Dimension:
Case:

Figure 1: Screenshot of LEd editor

Our view is that recursive feature structures do not belong in the database, which is primarily designed as a repository for primitive lexical data. Recursive feature structures will be constructed in a subsequent, morpho-syntactic layer of the system that is designed for dealing with phenomena such as morphological analysis, compound words, and syntax.

4.2. LEd

To address the goal of interacting with linguists, MaltiLex includes LEd, a lexicon editor. This is a client program operating over the internet which allows the linguist to create and maintain lexical entries. A key feature of LEd is that in order to minimise the amount of information that has to be supplied, the form displayed for collecting lexical information is generated dynamically on the basis of a previously declared system of lexical values.

Figure 1 shows a screen-shot of the lexical entry for the word *kantin* (cellar) being updated with lexical information.

The exact set of attributes and values that appear in the template for accepting values is a consequence of certain inherent dependencies between the values. So, for example, an entry with syntactic category noun will have case and number but not tense, whilst a verb will have number but not case.

To describe these dependencies we have developed a language called ODL (Object Description Language) as briefly described in the next section.

5. ODL

Given the onerous nature of hand crafting lexical entries, ODL is designed, first and foremost, to help the linguist by allowing redundancy of expression: the linguist writes less, and the machine, thanks to the ODL description, fills in the gaps. Although in some ways ODL shares assumptions with constraint-based linguistic formalisms not purport to have the same expressive power. The primary aim is to provide a descriptively efficient mechanism for basic linguistic information.

It is important to point out that although ODL is used to create descriptions that are clearly language-specific, ODL itself is language-independent. It could thus easily be used with languages other than Maltese, though to date no such undertaking has been carried out.
An ODL script comprises a set of declarations of different types. The three most important types of declaration are Enumeration, Class, and Rule.

5.1. Enumeration Declarations

The purpose of an enumeration declaration is to create a named type and associate it with a set of possible values. So for example

```
enum number = {sing, plur, dual}
gender = {masc, fem, neuter}
cat = {N,V,A}
```

creates the type `number` and associates it with the possible values `sing, plur` and `dual`.

5.2. Class Declarations

Classes correspond to specific collections of attributes that (i) can be represednted as a string of bits and (ii) can be used to drive the user interface by generating exactly the right attrbutes/values to be filled in.
A class groups together a set of attributes and assigns a set of possible values to them. The values assigned must be consistent with the type definitions.
For example, we can create the class Noun having category N with

```
class Noun
{ cat = N; }
```

Note that we cannot assign, for example, `fem` since this would contradict the type declaration. We can also assign *disjunctions* of value which are consistent with the type. For example, below we define `CS`.

```
class CS
{ type = common;
  number = *;
  gender = masc|fem; }
```

Here the `type` attribute is common, which is fully specified. The `number` attribute is completely unspecified as indicated by the asterisk. The `gender` attribute is either `masculine` or `feminine` which, the reader should note, is not fully specified but nevertheless more specified than the bare enumeration.
The first value, in this case `masc` is called a *default values* which is assigned when an object in that class is first created. Of course, this value can be be changed to `fem` since that is licenced by the class declaration.

5.3. Special Values: Unspecified, Undefined, and None

Sometimes, we wish to express that a value may be unspecified. In English, for example, an article like "their" is unspecified for gender. Similarly, in Maltese, the word "intelliġenti" (intelligent) is fully underspecified for gender and number, and can combine with singular, plural, masculine and feminine nouns. In ODL, this is handled using the value `unspecified`.
A related notion is that of being undefined, meaning that the attribute in question is *not applicable* to this item. For example, plural nouns in Maltese are not defined for gender, i.e. cannot have a grammatical gender specification. From the perspective of the user interface, this means that as soon as a noun is specified as plural, the text box denoting gender should become inactive by greying it out. It also means that gender will not figure as a attribute in any higher constituent that involves a plural noun. In ODL, this is handled with the value `undefined`.
Some attributes in words refers to optional elements/values, mainly clitics like the "-u" in "rasu" (his head), or the "-na" in "rana" (our head). These can be present or absent in a word. If they are absent, we use the value `none`.

5.4. Inheritance

We also allow *normal and multiple inheritance* between classes, so that with

```
class NCS: Noun, CS;
```

we can define `NCS` as a subclass of Noun and CS i.e. the class of common nouns. Attributes and values are inherited from both classes. There is no limit to the number of possible superclasses in the list. The order, however, is important, since conflicts are dealt with using order-dependent precedence rules.

5.5. Labels

Labels can be thought of as ready-made class templates that can be chosen by a user and applied to a class. They ere convenient for defining prearranged collections of attribute-value and are in fact used for defining tagsets.
For example, a typical tag might be NMSG, meaning masculine singular common noun. Given the class declaration above, we would declare this with respect to the class `N` as follows:

```
define NNSM on NCS {
  number=sing; gender=masc }
```

It is important to distinguish between a class declaration and a label definition. The former yields an internal structure which is capable of supporting *instances*. Such instances not only control the interface, but also generate the internal representation of lexical values.
In contrast, the latter is just shorthand for a lexical value, and as a consequence, can be used to provide values for every tag in the tagset. Some further examples are shown in section 7..

5.6. Rules

Rule declarations can express dependencies between combinations of attributes and values. In Maltese a noun that is plural does not carry gender. We can express this rule as follows:

```
if (Number == plur) { !Gender }
```

The effect of this rule, when executed, is to prevent gender ever being assigned to an instance of any class whose `Number` attribute is `plur`. Under these conditions the option to assign gender would never appear in the user interface.

6. Lexical Information for Maltese

This section describes a system of lexical information for Maltese[3], which is intended to serve three purposes. First of all, it must capture our linguistic intuitions regarding the subject matter at hand as faithfully as possible. Secondly, it should cause the user interface to behave in a way that minimises user effort. Finally, it should include definitions for all the tags employed by the tagger, to allow for the future possibility of automatic creation of new lexical entries as a by product of text-tagging.

6.1. Basic Attributes and Values

Lexical information is expressed in terms of the attributes category, type, person, number, gender, and attachment. The basic value domains for each of these the attributes are declared using the following enumeration declarations.

```
enum Category {
  noun, verb, pseudo_predicate,
  modifier, participle, determiner,
  pronoun, conjunction, numeral,
  interjection, other }
```

These are followed by allowable subtypes within those categories, as follows.

```
enum Type {
  common, proper, indicative,
  imperative, adjective, adverb,
  active, passive, quantifier,
  indefinite, demonstrative, subject,
  object, interrogative, coordinating,
  subordinating, cardinal_transitive,
  cardinal_intransitive, ordinal,
  interjection, negative_marker,
  preposition, reflexive, article,
  aspectual_marker }

enum Attachment {
  none, bound_t, clitic1, clitic2,
  bound_t_clitic1, clitic1_clitic2,
  plus_article, negative }
```

Finally, there is the following set of agreement features.

```
enum Person {
  first, second, third }

enum Number {
  sing, plural, dual,
  collective, unspecified }

enum Gender {
  masc, fem, unspecified }
```

[3]the linguistic content of this informtion is due to Ray Fabri

6.2. Classes

With the basic attributes and their values in place, we can now define the entry classes which, besides corresponding to types of lexical entry, also control the appearance of the user interface.

```
class Noun {
  Category   = noun;
  Type       =
    common | proper;
  Number     =
    sing | plural | dual |
    collective | unspecified;
  Gender     =
    masc | fem | unspecified;
  Attachment =
    none | bound_t | clitic1 |
    bound_t_clitic1 | undefined ;
}

class Verb {
  Category = verb;
  Type =   indicative | imperative;
  Person = first | second | third;
  Number = sing | plural;
  Gender =
    masc | fem | undefined;
  Attachment =
    none | clitic1 | clitic2
    | clitic1_clitic2 | negative;
}

class Pseudo_Predicate {
  Category = pseudo_predicate;
  Attachment = clitic1 | negative;
}

class Modifier {
  Category = modifier;
  Type = adjective | adverb;
  Number = sing | plural
           | unspecified | undefined;
  Gender = masc | fem
           | unspecified | undefined;
}

class Participle {
  Category = participle;
  Type = active | passive;
  Number = sing | plural;
  Gender = masc | fem;
}

class Determiner {
  Category = determiner;
  Type = quantifier| indefinite
         | demonstrative;
  Number = sing | plural
           | undefined;
  Gender = masc | fem
```

```
           | undefined;
  Attachment =
    none | plus_Article
    | undefined;
}

class Pronoun {
  Category = pronoun;
  Type = subject | object | demonstrative
         | interrogative | indefinite;
  Person = first | second | third
           | undefined;
  Number = sing | plural
           | undefined;
  Gender = masc | fem
           | unspecified | undefined;
}

class Conjunction {
  Category = conjunction;
  Type = coordinating | subordinating;
}

class Numeral
{
  Category = numeral;
  Type =     cardinal_transitive
             | cardinal_intransitive
             | ordinal;
}

class Interjection {
  Category = interjection;
  Type = interjection;
}

class Other {
  Category = other;
  Type =
    negative_marker | preposition
    | reflexive | article
    | aspectual_marker;
  Person =
    first | second | third | undefined;
  Number =
    sing | plural | undefined;
  Gender =
    masc | fem | undefined;
  Attachment =
    none | plus_article
    | clitic1 | undefined;
}

class CommonNoun: Noun; {
 Type = common }

class ProperNoun: Noun; {
 Type = proper }
```

7. Work in Progress: Tagset

The Maltese tagset was orginally developed by Gatt et al. (2003) for the purpose of tagging texts and with the aim of training an automatic tagger. This setting is somewhat different from that which gave rise to the lexical value system described in section 6. which has been designed for the purpose of adequately characterising the lexical properties of the Maltese language. In addition, these two pieces of developement work took place at geographically separated sites.
Separate development of these two elements has led to some incompatibilities, and we are currently in the process of bringing the two strands of work together. We are making use of the label declarations described above to bring this about. Below we give an example of how this will be done, restricting our attention to nouns.

```
define NNSM on CommonNoun {
 Number = sing; Gender = masc }

define NNSF on CommonNoun {
 Number = sing; Gender = fem }

define NND on CommonNoun {
 Number = dual; Gender = unspecified }

define NNP on CommonNoun {
 Number = plur; }

define NNSMC1 on CommonNoun {
 Number=sing; Gender=masc
 Attachment=clitic1 }

define NNSFC1 on CommonNoun {
 Number=sing; Gender=fem
 Attachment=clitic1 }

define NNPC1 on CommonNoun {
 Number=plur;
 Attachment=clitic1 }

define NNSF-G on CommonNoun
 {Number=sing; Gender=fem;
  Attachment = bound_t }

define NP on ProperNoun;
```

8. Conclusion and Future Work

We expect to have completed the mapping between the tagset and the lexical value system in the near future. Two main lines of development are then foreseen. Once the tagset issue is resolved, our next priority is the training of an accurate tagger, since we are convinced that the availability of large quantities of tagged text will open the way to a number of useful applications such as named entitiy extraction, automated text classification etc. A tagger and a lexicon that agree with each other also open the door to the possibility of automated extraction of lexical entries from corpora.

A second cycle of developement concerns syntax, which so far has been largely ignored. In order to develop a convincing syntactic description of Maltese, we plan to use HPSG, extending the analysis of a preliminary fragment of Maltese that has already been undertaken by (Müller, 2007). In this respect, it is of prime importance to investigate the possibilities of interfacing the HPSG lexicon with MaltiLex.

9. Acknowledgements

The work reported in this paper has been carried out in collaboration with colleagues from the University of Malta including in particular Ray Fabri, Duncan Attard, Mike Spagnol, and Albert Gatt. I am also indebted to the Malta Council for Science and Technology for supporting the MLRS project.

10. References

R.A Amsler. 1980. The structure of the merriam-webster pocket dictionary. Technical report, University of Texas at Austin.

B.K. Boguraev, T. Briscoe, J. Carroll, D. Carter, and C. Grover. 1987. The derivation of a gramaticaly indexed lexicon from the longman dictionary of contemporary english. In *Proceedings of the 25th Annual Meeting of the ACL, Stanford University*, pages 193–200.

Martin C. Emele. 1994. The typed feature structure representation formalism. *Proceedings of the International Workshop on Sharable Natural Language Resources*.

David Farwell, Louise Guthrie, and Yorick Wilks. 1992. The automatic creation of lexical entries for a multilingual mt system. In *Proceedings of the 14th conference on Computational linguistics*, pages 532–538, Morristown, NJ, USA. Association for Computational Linguistics.

C. Fellbaum, editor. 1998. *Wordnet, an Electronic Lexical Database*. MIT Press.

A. Gatt, A. Vella, and J. Caruana. 2003. Annotating textual and speech data in maltese. In *Technical Note ISO/TC 37/SC 4*. International Standards Organisation - Language Resource Management.

Ronald M. Kaplan and Joan Bresnan. 1982. Lexical-functional grammar: A formal system for grammatical representation. In *Joan Bresnan, editor, The Mental Representation of Grammatical Relations*, pages 173–281. MIT Press.

M. Kay. 1984. Functional Unification Grammar: A formalism for Machine Translation. In *Proceedings of Coling*, pages 75–78, Stanford University.

Adam Kilgarriff, Pavel Rychly, Pavel Smrz, and David Tugwell. 2004. The sketch engine. Technical Report ITRI-04-08, Information Technology Research Institute, University of Brighton. Also published in Proceedings of Euralex, Lorient, France, July 2004, pp. 105-116.

Kemps-Snijders Marc, Mark-Jan Nederhof, and Peter Wittenburg. 2006. Lexus, a web-based tool for manipulating lexical resources. Technical report, Max-Planck-Institute for Psycholinguistics Wundtlaan1, 6525 XD Nijmegen, The Netherlands.

Stefan Müller. 2007. The Grammix CD Rom. a software collection for developing typed feature structure grammars. In Tracy Holloway King and Emily M. Bender, editors, *Grammar Engineering across Frameworks 2007*, Studies in Computational Linguistics ONLINE, pages 259–266. CSLI Publications, Stanford.

C. J. Pollard and I. A. 1987 Sag. 1987. *Information-Based Syntax and Semantics: Volume I, Fundamentals*, volume Volume 13 of CSLI Lecture Notes. Center for the Study of Language and Information.

M. Rosner, J. Caruana, and R. Fabri. 1998. Maltilex: A computational lexicon for maltese. In M. Rosner, editor, *Computational Approaches to Semitic Languages: Proceedings of the Workshop held at COLING-ACL98, Université de Montréal, Canada*, pages 97–105.

M. Rosner, R. Fabri, D. Attard, and Albert Gatt. November 2006. Maltese Language Resource Server. In *Proceedings of CSAW06, University of Malta*, pages 90–98.

C. Tsalidis, A. Vagelatos, and G. Orphanos. 2004. An electronic dictionary as a basis for nlp tools: The greek case. In *Proceedings Traitement Automatique du Langage Naturel (TALN)*, Fès, Maroc.

P. Vossen. 1997. Eurowordnet: a multilingual database for information retrieval.