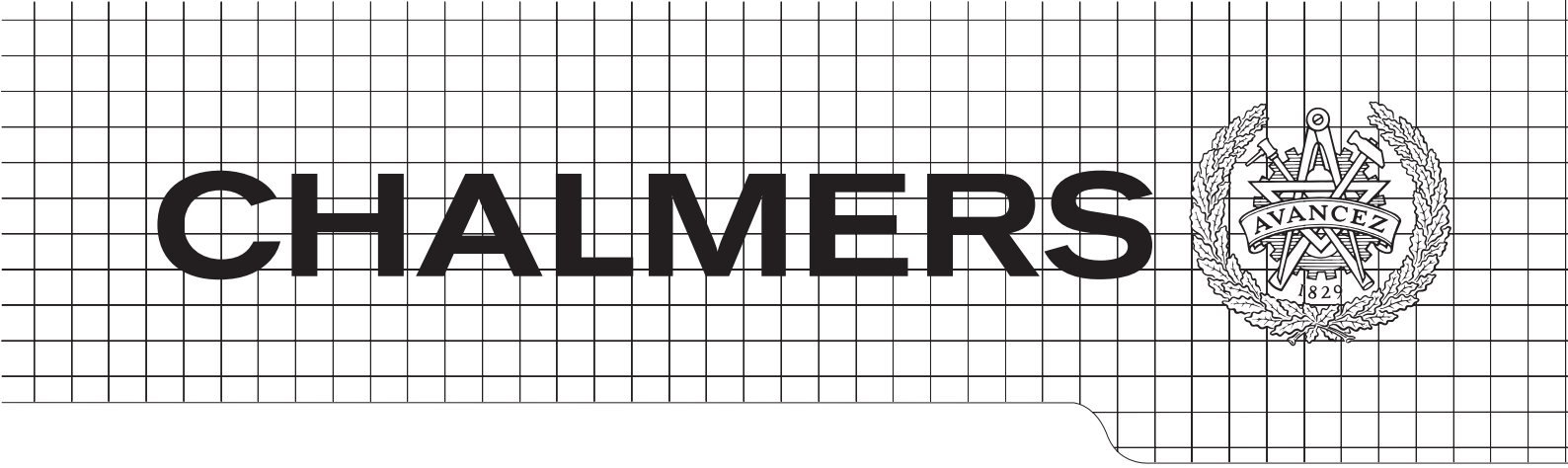


A Computational Grammar and Lexicon for Maltese

Master of Science Thesis in the programme:
Computer Science — Algorithms, Languages and Logic

John J. Camilleri

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
Göteborg, Sweden, September 2013

A Computational Grammar and Lexicon for Maltese
John J. Camilleri

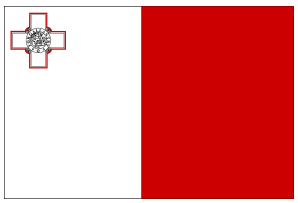
Examiner: Aarne Ranta

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone +46 (0)31-771 1000

Typeset in Palatino and Liberation Mono using XeLaTeX.

Department of Computer Science and Engineering
Göteborg, Sweden, September 2013

The research work disclosed in this publication is funded by the Strategic Educational Pathways Scholarship (Malta). The scholarship is part-financed by the European Union — European Social Fund (ESF) under Operational Programme II — Cohesion Policy 2007-2013, "Empowering People for More Jobs and a Better Quality of Life".



Operational Programme II — Cohesion Policy 2007-2013
Empowering People for More Jobs and a Better Quality of Life
Scholarship part-financed by the European Union European Social Fund (ESF)
Co-financing rate: 85% EU Funds; 15% National Funds
Investing in your future

This work was partly supported by the MOLTO project, via funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. FP7-ICT-247914.

Don't guess if you know.

Abstract

A Computational Grammar and Lexicon for Maltese

John J. Camilleri

Department of Computer Science and Engineering

Chalmers University of Technology and University of Gothenburg

Maltese is the national language of Malta and an official language of the European Union. While classified as Semitic, Maltese has been heavily influenced by the Romance languages and English, and features both root-and-pattern and concatenative morphologies. Despite its active use, the language is highly under-resourced in digital terms. This thesis contributes two computational resources for Maltese: a grammar and an online full-form lexicon.

The first part of this thesis deals with a computational grammar for Maltese, which is implemented using the Grammatical Framework (GF). GF is a multilingual grammar formalism based on using abstract syntax trees as language-independent semantic representations. Its Resource Grammar Library (RGL) already covers the morphology and basic syntax of some 27 languages from around the world. Maltese is the 28th addition to the RGL, and the first Semitic language in the library to be completed. The smart paradigms implemented in the morphological part of grammar allow full inflection tables to be produced for any lexical unit, often requiring only a lemmatised form. This report looks at some of the more interesting implementational details of the grammar, discussing the compromises that had to be made along the way.

The second part covers the collection of various Maltese lexical resources into a single searchable collection, using a schema-less database to accommodate partial data from heterogeneous sources. We then use the smart paradigms from the morphological part of the grammar to automatically produce some 4 million inflection forms and extend the collection into a full-form computational lexicon, which can be used in for morphological lookup and spell checking.

All the software and resources described in this thesis are open-source and free to use for any purpose.

Keywords: *computational, grammar, syntax, morphology, lexicon, linguistics, Maltese*

Acknowledgements

This thesis is in many ways the indirect product of countless people who have inspired and guided me in my academic career so far. However, a few people in particular deserve special thanks. They are:

- My supervisor Aarne Ranta, not just for getting me into GF but for often seeing more in me than I did myself.
- All the GF community, in particular Krasimir Angelov and Thomas Hallgren for their patience with answering all my silly questions in the beginning, and all my tougher questions later on.
- Michael Spagnol and others from the Maltese linguistics community for eagerly sharing their resources with me and providing a much-needed linguist viewpoint.
- Albert Gatt and Claudia Borg for encouraging collaboration, trusting me with the keys to the MLRS and offering their highly valued feedback on much of what I have done.
- My undergraduate supervisors Mike Rosner and Gordon Pace, for getting me into computational linguistics in the first place, showing me the ropes of academic research and linking me to the Language Technology group here at Chalmers.

Finally, I would not and could not have gotten this far without the support of my fiancée Claudia, who traded sun and sea for cold and dark just so that I could pursue my ambitions.

Contents

1 Introduction and background **1**

1.1 The Maltese language . . . 1

1.1.1 History . . . 1

1.1.2 Overview . . . 2

1.1.3 Maltese and the digital age . . . 4

1.2 Linguistic tools and resources for Maltese . . . 5

1.2.1 Print resources . . . 5

1.2.2 Corpora and terminologies . . . 6

1.2.3 The Maltilex project . . . 7

1.2.4 MLSS . . . 8

1.2.5 Other resources . . . 8

1.3 The Grammatical Framework . . . 9

1.3.1 Multilingual grammars . . . 9

1.3.2 The GF runtime . . . 13

1.3.3 Resource Grammar Library . . . 14

1.3.4 Lexical resources . . . 15

1.4 Goals . . . 16

1.4.1 Motivation . . . 16

1.4.2 Contributions . . . 16

1.4.3 Organisation . . . 17

2 Resource grammar **18**

2.1 Module structure . . . 18

2.2 Morphology . . . 19

2.2.1 Smart paradigms . . . 19

2.2.2 Verb . . . 20

2.2.3 Noun . . . 25

2.2.4 Adjective . . . 27

2.2.5 Other lexical categories . . . 28

2.2.6 Euphonic vowels . . . 29

2.3 Syntax . . . 30

2.3.1 Particles . . . 30

2.3.2 Numerals . . . 32
2.3.3 Pronouns . . . 34
2.3.4 Tense-anteriority system . . . 35
2.3.5 Phrases . . . 36
2.3.6 Clauses . . . 38
2.4 Development and testing . . . 39
2.4.1 Development tools . . . 39
2.4.2 Test methodology . . . 40
2.4.3 Treebanks . . . 41
2.4.4 Coverage . . . 42
2.5 Problems encountered . . . 43
2.5.1 Large inflection tables . . . 43
2.5.2 Binding of runtime tokens . . . 44
2.5.3 Pattern matching on runtime strings . . . 44
2.5.4 Non-existent forms . . . 44
2.5.5 Parameters in tables . . . 45

3 Computational lexicon **47**
3.1 Method . . . 47
3.1.1 Sources . . . 47
3.1.2 Heterogeneous data . . . 47
3.1.3 Full-forms . . . 48
3.2 Application . . . 50
3.2.1 Database . . . 50
3.2.2 Web application . . . 53
3.2.3 Web service . . . 54
3.2.4 Issues . . . 54
3.3 Monolingual GF dictionary . . . 56
3.3.1 Method . . . 56
3.3.2 Dictionary modules . . . 57
3.4 Generating full forms . . . 58
3.4.1 Nouns . . . 58
3.4.2 Adjectives . . . 58
3.4.3 Verbs . . . 58

4 Conclusions **60**
4.1 Contributions . . . 60
4.2 Limitations . . . 60
4.2.1 Grammar limitations . . . 60
4.2.2 Lexicon limitations . . . 61
4.3 Related work . . . 62

4.3.1 Computational linguistics resources for Maltese . . . 62
4.3.2 GF resource grammars . . . 63
4.3.3 Grammar formalisms . . . 63
4.3.4 Rule-based translation . . . 64
4.4 Future work . . . 64
4.4.1 Grammar extensions . . . 64
4.4.2 Lexing solution . . . 65
4.4.3 A linked Maltese WordNet . . . 65

A Resource grammar structure **66**
A.1 Categories . . . 66
A.2 Modules . . . 68
A.3 Paradigms . . . 69

B Lexicon analyses **74**
B.1 Analysis of noun paradigms . . . 74
B.1.1 Determinate and indeterminate plurals . . . 74
B.2 Analysis of verb participles . . . 79
B.3 Vowel changes . . . 82
B.3.1 Object suffixes . . . 82
B.3.2 Vowel length and negation . . . 83

C Inflectional forms of a Maltese verb **84**

D Links, source code and licenses **100**

Glossary **101**

Bibliography **104**

List of Tables

2.1 Verb classes and their characteristics . . . 20
2.2 Inflection table for Semitic verb *ħareġ* ('he went out') . . . 21
2.3 Derivational forms for triliteral verb roots . . . 22
2.4 Derivational forms for quadriliteral verb roots . . . 22
2.5 Inflection table for an English-loan verb *pparkja* ('he parked') . . . 22
2.6 Enclitic pronouns on the verb *xtara* ('he bought') . . . 23
2.7 Noun numbers . . . 25
2.8 Noun paradigms . . . 25
2.9 Morphological processes for plurals . . . 26
2.10 Noun paradigms . . . 27
2.11 Inflection table for adjectives . . . 28
2.12 Adjective paradigms . . . 28
2.13 Examples of lemmas with and without euphonic vowels . . . 29
2.14 Behaviour of euphonic vowels in various situations . . . 30
2.15 Assimilation of particles . . . 31
2.16 Enclitic pronouns . . . 34
2.17 Maltese tenses . . . 35
2.18 RGL tense-anteriority system and correspondences to Maltese tenses . . . 36
2.19 Examples of variable word order . . . 38
2.20 Treebank list and testing results . . . 42

3.1 Size estimation of a full-form lexicon . . . 50
3.2 Example database queries and response times . . . 55

A.1 Category list . . . 67
A.2 Concrete module summary . . . 68
A.3 Resource module summary . . . 69
A.4 Morphological paradigms: constructors . . . 69
A.5 Morphological paradigms: nouns . . . 70
A.6 Morphological paradigms: verbs . . . 71
A.7 Morphological paradigms: others . . . 72

B.1 Analysis of noun plural forms . . . 75

B.2 Analysis of verb participles 79
B.3 Corpus study of vowel length changes under negation 83

C.1 Full inflection table for the verb *fetaħ* ('he opened') 84

List of Figures

1.1 State of language technology support for Maltese 5
1.2 Abstract syntax trees for the Foods grammar 10

2.1 Screenshot of the RGL browser tool 39
2.2 Screenshot of the syntax tree editor 40

3.1 Basic database schema 51
3.2 Screenshot: searching by root 53
3.3 Screenshot: searching by lexeme 54
3.4 Screenshot: viewing an entry's details 55

A.1 Dependencies of phrasal and lexical categories 66
A.2 Module dependency graph 73

Conventions

The following conventions are used in this thesis:

- Maltese words, affixes, and phrases are always shown in italics: *ġemel*, *-ijiet*, *kliem ix-xiħ*.
- English glosses for Maltese words or phrases are shown within single quotes after the entity which they explain: *ssogra* ('he risked').
- The Maltese verb is represented by the third person singular masculine in the perfective ("*mamma*"), since it is morphologically unmarked. There is no infinitive in Maltese.
- Tri-consonantal and quadri-consonantal Semitic roots are shown preceded by the root symbol: √FTĦ, √GRBB.
- Inline code snippets or identifiers from source code are written in `monospace`.

A glossary of abbreviations and acronyms can be found on page 101.

Chapter 1

Introduction and background

We begin by looking at the Maltese language, and the computational linguistics research that exists for the language. We then continue to give a background of the Grammatical Framework and its Resource Grammar Library. Finally we look at the motivations for this work and its contributions to the field.

1.1 The Maltese language

This section contains a brief history of Maltese and its general characteristics, along with general account of its use today. More in-depth linguistic details of the language are covered in chapter 2.

1.1.1 History

Due to their central location in the Mediterranean sea, the Maltese islands have had a long history of colonisation. Starting from the eighth century BCE, the islands have passed through the hands of the Phoenicians, Romans, Byzantines, Normans and the Aragonese. Yet it was the Arabs, who conquered the archipelago in 870 AD, who left the first lasting linguistic mark on the islanders. Probably imposing their language on the locals, their influence became even more marked in 1048 when the rulers brought to the islands a new community which spoke Siculo-Arabic. This period marks the beginning of what we today call Maltese.

When the Normans took the islands in 1090 and annexed them to the Kingdom of Sicily, most contact with Arab world was lost and a process of Latinisation was begun. Contact with Romance languages only increased when in the 16th century the archipelago was passed to the Knights Hospitaller of St. John — most of whom were French and Spanish, but used Italian as the written language of culture and administration.

The 19th century brought new rulers to the islands — the British, whose attempts to Anglicise the country led to the 'Language Question', a linguistic and cultural conflict which eventually replaced Italian (the primary written language of the islands) with English. This conflict did however enhance the role of Maltese, which together with English was made an official language of Malta in 1934. Malta became independent from British in 1964 and become a republic ten years later, though English is still retained as an official language alongside Maltese.

Maltese today is still Semitic in origin, but differs considerably from other neo-Arabic languages as it has been heavily influenced by Romance languages and English. It is the national language of the Republic of Malta, and its official language together with English. Since 2004, Maltese has also been an official language of the European Union. Maltese is spoken by 90% of the roughly 400,000 inhabitants of the Maltese islands, as well as by some smaller expatriate groups around the world.

1.1.2 Overview

This section provides a very brief overview of the Maltese language. For complete, in-depth descriptions of the language refer to Azzopardi (2007), Brother Henry F.S.C. (1980) and to Spagnol (2011) for an in-depth look at morphology.

Alphabet

Maltese is the only Semitic language to use a Latin alphabet, with some minor additions. The complete alphabet consists of 30 letters (including two digraphs), split into 6 vowels and 24 consonants:

a b ċ d e f ġ g għ h ħ i ie j k l m n o p q r s t u v w x ż z

The vowels in Maltese are: *a e i ie o u*. Vowels at the ends of words sometimes take the grave accent, for example *università* ('university') and *Ġesù* ('Jesus'). The circumflex can used to indicate vowel length as in *sâb* ('he found'), but is in general omitted in standard orthography.

All consonants are considered **strong/sound** (*sħaħ*), with the exception of *j* and *w* which are **weak** (*dgħajfin*). The **coronal consonants** (*konsonanti xemxin*; opposite: *qamrin*) are the subset which assimilate with the article: *ċ d n r s t x ż z*. The **liquid consonants** (or **sonorants** (Fabri, 2009)) are those which assimilate with a prefixed 'n': *l m n r*. Together with *għ*, the liquids require the addition of epenthetic vowels during conjugation.

Morphology

Nouns inflect for number, which can be singular, dual, collective, or plural (though all forms tend not to exist together). Animate nouns also inflect for gender (e.g. *spiżjar/spiżjara*, 'male/female pharmacist'), while others have an intrinsic gender. Plurals may be either **sound** (external) or **broken** (internal); some nouns have both, e.g. *tapit* ('carpet') → *tapiti*, *twapet* ('carpets').

Adjectives behave similarly to nouns, but have only a singular and plural (which is often broken as in the case of nouns). They have positive form, a comparative which may be formed either morphologically or syntactically, and a superlative obtained by prefixing the definite article: *kbir* ('big'), *ikbar* ('bigger'), *l-ikbar* ('the largest').

Verbs inflect for tense (mood/aspect), person (P1 sg/pl, P2 sg/pl, P3 sg m/f, P3 pl) and polarity. They often take enclitic pronouns as direct and indirect objects. Many verbs have a

past/passive participle, while fewer also have a present/active participle. Up to 9 derivational processes exist for root-and-pattern verbs which, typically producing passive or reflexive versions of the original verb.

Maltese morphology is based on two distinct yet often combined systems: **root-and-pattern morphology**, which is a decidedly Semitic feature, and **concatenative morphology**, which in general comes from the Romance and English influences on the language.

Root-and-pattern morphology This type of morphological centres around the idea of a **root**, a sequence of three or four consonants known as radicals which form some kind of morphological unit. Roots are generally classified according whether they are tri- or quadri-consonantal, and whether any of the radicals is a weak consonant (see section 2.2.2). A **pattern** is a template which describes how the radicals in a root can be combined with a vowel sequence to produce a word form of a specific lexical category and with specific inflectional features.

Thus the pattern C1vC2vC3 dictates that when applied to a tri-consonantal root and suitable vowel sequence, we can produce a verb of binyan 1 in the perfective aspect, inflected for P3 m. sg. For example, under this pattern the root $\sqrt{\text{KTB}}$ and vowel sequence *i-e* form the inflected verb *kiteb* ('he wrote').

The radicals of a root may never appear in any other order, yet they may or may not have vowels or constant consonants (consonants which are part of the pattern) between them. Note how this kind of morphology is discontinuous; there is no concept of a word stem which is extended with affixes. Root-and-pattern morphology is capable of describing both inflection and derivational processes together.

Concatenative morphology The root-and-pattern system is not responsible for all morphological processes however; Maltese morphology is also rife with concatenation. The imperfective verb paradigm for example is formed by prefixation of consonants *n*, *t* and *j* to a stem form: *nikteb* ('I write'), *tikteb* ('you (sg.) write'), *jikteb* ('he writes').

Concatenation is all the more pronounced in the inflection of **loan words** — words of non-Semitic origin. While there exist a number of non-Semitic words which have been integrated into the root-and-pattern system (types A and B in Mifsud (1995)), this list is now closed and no longer productive (Spagnol, 2011, p. 44.). Instead, all new loans brought into Maltese have a purely concatenative morphology (Mifsud's types C and D). Examples include the verb *sfida* ('he defied') from the Italian 'sfidare': *sfidajt* ('I defied'), *sfidat* ('she defied'), *sfidajna* ('we defied'); and the noun *kejk* from the English 'cake', which is suffixed in the plural: *kejkijiet* ('cakes').

Syntax

Maltese displays the typical properties of an SVO language, yet its word order is highly flexible. Constituent orders SVO, VOS, and OVS are all possible and have an emphasising affect, and accompanied by a distinct intonation when spoken. Similarly, questions are typically VOS but can be re-ordered in various ways for effect.

The definite article is the proclitic *il-*, which becomes *l-* in front of vowels and where the *l* changes to match succeeding coronal consonants: *il-* + *dar* → *id-dar* ('the house'). The article also joins with prepositions, *ma* + *il-* + *fornar* → *mal-fornar* ('with the baker'). It is invariant for gender, number and used for both both nouns/adjectives. Absence of the article gives the indefinite form.

Except for the numeral *wieħed* ('one'), all cardinal numerals precede the noun they quantify. This is true for all other quantifiers and demonstratives, which are located to the left of the noun which has to host the definite article if the demonstrative is used: *dan* + *il-* + *ajruplan* → *dal-ajruplan* ('this aeroplane'). Proper nouns are exempt from this rule.

The pronouns available in Maltese are similar to those as in English. Pronouns are often enclitic, indicating possession when joined with nouns (e.g. *missier* + *tiegħi* → *missieri*, 'my father'), and as direct and/or indirect objects when attached to verbs (e.g. *ħareġ* + *aħna* → *ħar-iġna*, ('he took us out')). Pronouns are only used as topic or for emphasis; otherwise Maltese is generally a pro-drop language. Maltese has only a single relative pronoun *li* which is common for all persons, genders and number.

Verbs have two moods: **indicative** and **imperative** (there is no infinitive in Maltese). The indicative consists of two aspects: **perfective** for actions that have been completed, and **imperfective** for those which are ongoing. Other tenses are formed by combining with the auxiliary verb *kien* ('he was') and other particles such as *ma* for the negative.

(Rosner & Joachimsen, 2012; Stolz, 2011)

Vocabulary

Given the heterogeneous makeup of Maltese, it is no surprise that its vocabulary is also quite varied. A study of the etymological sources of Aquilina's dictionary surprisingly found that the majority of lexical entries in Maltese are in fact Siculo-Italian (52%), followed by Semitic (32%) and English (6%). The remaining percentage can be accounted for by purely local formations and lexical items with unknown etymologies.

On the other hand, type vs. token frequency counts in journalistic texts have revealed that almost 73% of items are of Semitic origin. This can be explained by the fact that prepositions, pronouns, and other grammatical words are mostly derived from Arabic. (Spagnol, 2011, p. 12)

1.1.3 Maltese and the digital age

A recent META-NET report on the language (Rosner & Joachimsen, 2012) indicates that while a number of linguistic resources do exist for the languages, Maltese is in general under-represented in digital terms when compared to other European languages. As shown in figure 1.1, the study finds that Maltese is best resourced with speech synthesis and text corpora, but very poorly represented in other areas of computational linguistics. Specifically, to date there are no computational grammars, morphological analysers, or text generation systems for Maltese.

A Eurobarometer report (European Commission, 2011) found that in Malta, 89.5% of the respondents claimed that Maltese was their mother tongue. However when it comes to Internet

	Quantity	Availability	Quality	Coverage	Maturity	Sustainability	Adaptability
Language Technology: Tools, Technologies and Applications							
Speech Recognition	0.8	0.8	0.8	0.8	0.8	0.8	0.8
Speech Synthesis	2.4	0.8	3.2	3.2	2.4	2.4	2.4
Grammatical analysis	0.8	0.8	0.8	0.8	0.8	0.8	0.8
Semantic analysis	0	0	0	0	0	0	0
Text generation	0	0	0	0	0	0	0
Machine translation	1.6	1.6	1.6	1.6	1.6	1.6	1.6
Language Resources: Resources, Data and Knowledge Bases							
Text corpora	3.2	3.2	2.4	2.4	2.4	3.2	3.2
Speech corpora	2.4	0.8	2.4	1.6	2.4	2.4	2.4
Parallel corpora	3.2	3.2	2.4	1.6	1.6	1.6	1.6
Lexical resources	2.4	2.4	1.6	2.4	2.4	2.4	2.4
Grammars	0	0	0	0	0	0	0

Figure 1.1: State of language technology support for Maltese (scale from 0–9). Taken from Rosner & Joachimsen (2012).

usage, only 6.5% use exclusively Maltese on the Internet when reading, consuming content or communicating. An overwhelming 90.6% choose to browse websites in English and 20.1% in Italian, respectively.

The digital age has also seen a sharp rise in words loaned from English, in particular for Internet and social network concepts for which no desirable Maltese equivalent exists. The traditional principle for loan words is to write them according to Maltese orthographical rules, such as *imejl* for 'email'. However, many prefer to write English loan words with their original spelling. The habits of a language community and speed with which new technical words are loaned mean that the official bodies have a hard time deciding on — and introducing — new guidelines successfully (Il-Kunsill Nazzjonali ta' l-Ilsien Malti, 2008).

1.2 Linguistic tools and resources for Maltese

1.2.1 Print resources

The major paper dictionaries available are Ġużé Aquilina's Maltese-English-Maltese volumes (1987–1990) and Erin Serracino-Inglott's 10-volume *Il-Miklem Malti* (1975). Mario Serracino-Inglott's more recent Maltese-only dictionary and thesaurus (Serracino-Inglott, 2003) is most used in this work.

The most recent grammar we have is Borg & Azzopardi-Alexander (1997)'s *Maltese*, which is also the most-consulted reference in this work. Other print grammars which were not access-

ible to the author (and thus not listed in the bibliography) are Lawrenz Cachia's *Grammatika Ġdida tal-Malti* (1994) and Edmund Sutcliffe's *A Grammar of the Maltese Language* (1936). Older historical grammars include *A Short Grammar of the Maltese Language* by an unknown author in 1845, Francis Vella's *Maltese Grammar for the use of the English* (1831), and Michelantonio Vassalli's *Grammatica della lingua Maltese* (1827).

1.2.2 Corpora and terminologies

MLRS corpus

A monolingual corpus for Maltese has been compiled and is hosted by the Maltese Language Resource Server (MLRS)[1] (Rosner *et al.* , 2006). The MLRS Corpus (Gatt & Borg, 2011) can be described as an opportunistic text collection of nearly 130 million tokens, mostly created from publicly available documents, but also including a limited amount of user-contributed material. Texts for the version 2.0 BETA of the corpus were pre-processed to remove duplicate material and long stretches of non-Maltese text. They were also processed with a simple dictionary-based spelling correction, and part-of-speech (POS) tagged using the TnT Tagger trained on circa 26k of manually annotated text.

Parallel corpora

There are also some parallel corpora for Maltese, which have come from the translation efforts of the European Commission. The JRC-Acquis corpus (Ralf *et al.* , 2006) is a parallel corpus containing the complete text of the European Union Law (*Acquis Communautaire*) in 22 languages.

Other parallel texts for Maltese are accessible through the OPUS project (Tiedemann, 2009) — a open-access collection of translated texts from the web, automatically processed and annotated using open-source products. OPUS contains parallel corpora for English-Maltese from:

- EMEA - PDF documents from the European Medicines Agency (11.2M tokens)
- ECB - Website and documentation from the European Central Bank (2.7M tokens)
- EUconst - A parallel corpus collected from the European Constitution (0.1M tokens)

MAMCO — Maltese Multimodal corpus

Work has also begun on a Multi-modal corpus for Maltese, beginning with video recordings of first-encounter conversations between pairs of Maltese speakers. The setting and organisation replicate those used in the Nordic NOMCO corpus. Corpus annotations include transcription of the spoken data, as well as annotation of head movements and other gestures. This corpus has allowed studies on, amongst others, lengthening as a discourse strategy (Paggio *et al.* , 2013).

[1] http://mlrs.research.um.edu.mt/, accessed 2013-09-05

InterActive Terminology for Europe (IATE)

The IATE[2] is the European Union's multilingual term base used by various institutions for the collection, dissemination and shared management of EU-specific terminology. It was launched in 1999 as a web-based infrastructure for all EU terminology resources. IATE incorporates and standardises all existing terminology databases of the EU's translation services into a single database. It also includes a number of legacy databases, and now contains a total of approximately 1.4 million multilingual entries or 8.4 million terms, covering all 23 languages of the EU (including Maltese).

1.2.3 The Maltilex project

There has long been an interest in creating a computational lexicon for Maltese, though to date no concrete system yet exists. The Maltilex project was first announced in 1998, identifying the need for such a lexicon and outlining the scope of the project for creating one (Rosner *et al.* , 1998). Rosner *et al.* (1999) go on to highlight the automatic extraction idea, involving tokenising and performing headword identification on a corpus in order to obtain a lexicon.

In his M.Sc. thesis, Dalli (2002a) describes a concrete implementation of this, using a weakly-supervised learning approach. The work details at length the machine learning algorithms enabling extraction this, adapting clustering techniques from bio-informatics. It also introduces the Lexicon Structuring Technique (LST), which attempts to identify lemmas in an unstructured list of words without requiring any prior rules. Sadly, the only Maltese corpus available at the time was very small (~2000 tokens) and the end results were very noisy and not practically usable. Despite the corpus size having today grown to 100 million words, it seems this experiment has not been carried out again with the new larger corpus.

A later student project by Attard (2005) concentrates on the infrastructure required for implementing such a lexicon as a collection of services. Despite going into significant technical detail, the framework suffered from lack of flexibility and was not adopted in any lasting way by the project.

As the Maltilex project evolved into the Maltese Language Resource Server (MLRS), a new description of a lexicon structure was presented in Rosner *et al.* (2006). This paper describes the use of an Object Description Language (ODL) for the specification of the attributes and values that make up each lexical class, effectively acting as a kind of type system for a set of key-value pairs. However it is not clear that such a model has actually been written in ODL for Maltese. The paper only briefly treats the implementation of the lexicon database itself, noting only how the relational model is not entirely suitable and that no satisfactory storage format had yet been decided upon.

The most recent development in this road towards a computational lexicon is the announcement of a project to create a national online dictionary for Maltese[3]. Rather than focusing on extraction from a corpus, the project will instead be digitising Aquilina's Maltese-English dic-

[2] http://iate.europa.eu/iatediff/brochure/IATEbrochure_MT.pdf, accessed 2013-08-21
[3] National Council for the Maltese Language. http://kunsilltalmalti.gov.mt/projects, accessed 2013-06-24

tionary and making it available online. At the time of writing, the project is still in the process of updating the original dictionary to include modern entries and had not yet entered the digitisation phase (personal communication, July 2013).

1.2.4 MLSS

As part of the Metanet4U initiative, the Maltese Language Software Services (MLSS)[4] portal has recently made available a number of language-processing tools for Maltese. These tools cover the following tasks:

- paragraph and sentence splitting
- tokenisation (splitting running text into individual tokens)
- part of speech tagging (categorising each token in running text with its part of speech)

1.2.5 Other resources

Maltese root-and-pattern verb database

The online database of root-and-pattern verbs (Camilleri & Spagnol, 2013) is based on the exhaustive enumeration of 1,923 roots and over 4,142 verbs by Spagnol (2011). The original data was converted from Excel format into a relational database, with a web-based interface then built around it. This interface provides the ability to search for roots and verbs quickly using regular expression syntax. In addition to information on roots, classes and derived forms from the original source, fields for English gloss, transitivity information, and verb frequency were also added. The database is hosted on the Maltese Language Resource Server (MLRS)[5] and provides both a user interface and a JSON web service.

Corpus of broken plurals

Mayer *et al.* (2013) have put together a large corpus of broken plurals in Maltese, containing 654 singular-broken plural pairs mainly based on a list of synchronically used or known broken plurals compiled by Schembri (2012). The study is based on a comprehensive inventory of broken plural forms which an average educated speaker of Maltese would be expected to have. In cases of a broken plural having multiple variants, only the most common or frequent is given. The corpus also includes a small number of collective forms, such as *basal* ('onions') and *ġebel* ('stones'), which share the same CV structure of broken plurals but which differ morpho-syntactically. It is available in CSV format from the MLRS[6] website.

[4] http://metanet4u.research.um.edu.mt/, accessed 2013-05-03
[5] http://mlrs.research.um.edu.mt/resources/verbalroots/, accessed 2013-05-03
[6] http://mlrs.research.um.edu.mt/index.php?page=33, accessed 2013-08-14

Collection of verbal nouns

In her forthcoming M.A. thesis, Ellul (2013) has collected a list of over 2,500 verbal nouns from the Aquilina dictionary and extended it to include some new ones from other sources.

Basic English-Maltese dictionary

This bilingual word list was originally written by Grazio Falzon in 1997[7], consisting of alphabetically ordered English lemmas with their Maltese translation and pronunciation. It contains some 5458 English entries and includes gender information for nouns. The original source did not use the correct Maltese characters. The collection has since been updated and released as a TEI-compliant XML dictionary file within the META-SHARE project[8]

1.3 The Grammatical Framework

The Grammatical Framework (GF) (Ranta, 2011) is a functional language specialised for multilingual grammar applications. GF differs from unification-based grammar formalisms by having separate abstract/concrete syntax rules, a strong type system, and inherent support for multilingual grammars. GF grammars are declarative in nature, with a primary focus on the linearisation of syntax trees.

When implementing a computer language it is common to make the distinction between *abstract* and *concrete* syntax. The abstract syntax defines the language's hierarchical structure using trees, while the concrete syntax dictates what the language looks like as a string. This separation is based on the idea that type checking and semantics are more relevant to the abstract level, while syntax details are more a concrete concern. GF applies this distinction to natural languages too. Abstract syntax in GF is used as a language-independent interlingua for semantic representation, shared between multiple concrete syntaxes which describe the linearisation into different languages. By writing an abstract grammar and defining *how* it should linearise into one or more natural languages (concrete grammars), GF is able to derive both a generator *and* a parser for each of those languages. In this way, GF becomes a multi-lingual authoring and translation system which avoids the need to write separate rules for each language pair.

1.3.1 Multilingual grammars

Here we look at a small example of a GF grammar which is able to produce phrases about some foods. The grammar with two concrete syntaxes (English and Maltese) to demonstrate the separation between the language-independent concepts and

[7] Original URL: http://aboutmalta.com/language/engmal.htm, accessed 2013-08-22

[8] http://metashare.metanet4u.eu/repository/browse/basic-english-maltese-dictionary/13fc5802abc511e1a404080027e73ea2a210be7dd5c44a3b9dd47afb4b2a34ef/, accessed 2013-08-22

Abstract syntax

At the language-independent level we want to model semantically the kinds of comments we wish to make about food. Note how the names of categories and functions are only given in English for ease of understanding; they are merely identifiers and could be named anything. We start by defining the kinds of food in our grammar (`Cheese` and `Fish`) and the qualities they may have (`Expensive` and `Delicious`). The `Very` function exists for intensifying qualities, and `This` and `That` are functions which produce a demonstrative from a kind of food. Finally, the `Pred` (predication) function joins an `Item` and a `Quality` to give us a statement. Note that `Comment` is the top-level category in the grammar.

```
abstract Foods = {
  flags
    startcat = Comment ;
  cat
    Comment ; Item ; Kind ; Quality ;
  fun
    Cheese, Fish : Kind ;
    Expensive, Delicious : Quality ;
    Very : Quality -> Quality ;
    This, These : Kind -> Item ;
    Pred : Item -> Quality -> Comment ;
}
```

Given this abstract grammar, we can construct a some abstract syntax trees which use these functions (illustrated in figure 1.2):

```
Pred (This Fish) Delicious
Pred (These Cheese) (Very Expensive)
```

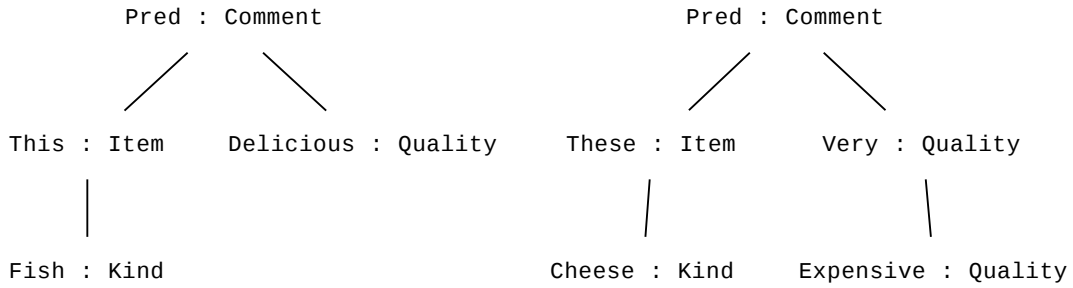


Figure 1.2: Abstract syntax trees for the Foods grammar

English concrete syntax

Now we want to describe how such trees can be linearised into our first language, English. We do this by writing a concrete grammar which contains `lincat` definitions for each of our

categories and `lin` rules for each function.

After defining a parameter `Number` which can either be singular or plural, we go on to say that a `Kind` can inflect for number. So it's linearisation type is a record with a single field `s`, which has the type of a table from `Number` to string[9]. In this was both singular and plural forms are carried around by the `Kind` instance until one of them is projected from the record. In the case of `Fish`, we specify that the surface form should always be 'fish' regardless of the number.

In English, adjective do not inflect for number, so we only need to store one string when it comes to the `Quality` category. The `Very` function works by pre-pending the token "very" to the token list projected from the function argument (`quality.s`). In the case of the determiner `Item`, we need to select the correct number form from the function argument `kind`, which is done using the bang operator `!`. We also add a new field `n` to the record which specifies the number of the demonstrative. Note this is different from inflecting for number, as we saw in the case of `Kind` above.

Finally, the `Pred` puts it all together by joining the tokens from its two arguments with a copula in between. `copula` is defined as an operator, which is a function of convenience that has no counterpart in the abstract syntax. In fact, all `oper` definitions are inlined during the compilation phase. In the case of English, the copula inflects only for number, and the `Pred` function takes care of selecting the copula form which agrees with the number of the `Item` argument.

```
concrete FoodsEng of Foods = {
  param  Number = Sg | Pl ;

  lincat Kind = { s : Number => Str } ;
  lin
    Cheese = { s = table { Sg => "cheese" ; Pl => "cheeses" }};
    Fish   = { s = table { _  => "fish" }} ;

  lincat Quality = { s : Str } ;
  lin
    Expensive = { s = "expensive" } ;
    Delicious = { s = "delicious" } ;
    Very quality = { s = "very" ++ quality.s } ;

  lincat Item = { s : Str ; n : Number } ;
  lin
    This  kind = { s = "this"  ++ kind.s ! Sg ; n = Sg } ;
    These kind = { s = "these" ++ kind.s ! Pl ; n = Pl } ;

  lincat Comment = { s : Str } ;
  lin  Pred item quality = {s = item.s ++ copula ! item.n ++ quality.s} ;
  oper copula : Number => Str = table { Sg => "is" ; Pl => "are" } ;
}
```

[9] `Str` in GF actually represents a token list, but we will use "string" as a synonym for this.

Maltese concrete syntax

For the Maltese concrete syntax of this grammar, we follow the same approach as in the English but have a few more agreement features to handle. We start by adding the `Gender` parameter, and then defining a few convenience operations for constructing tables from numbers and genders to strings. Note that we completely omit type declarations for these operators; the GF compiler will be able to infer their types.

The linearisation definition for `Kind` is similar in that it inflects for number, however we also add a static `g` field to hold the gender. Adjectives in Maltese inflect for both number and gender, so the `Quality` linearisation type contains a table from both of these to a string. The `numGenTbl` operation takes care of building this table for us. `Very` works in a similar way as in English, except that we place the modifier after the adjective and have the number and gender arguments to pass along. The rest of the grammar should also be quite understandable by now; once the parameters and linearisation types have been decided, the rules themselves tend to fall into place.

```
concrete FoodsMlt of Foods = {
  flags coding = utf8 ;
  param Number = Sg | Pl ;
  param Gender = Masc | Fem ;

  oper
    numTbl : Str -> Str -> (Number => Str) =
      \s,p -> table { Sg => s ; Pl => p } ;
    genTbl : Str -> Str -> (Gender => Str) =
      \m,f -> table { Masc => m ; Fem => f } ;
    numGenTbl : Str -> Str -> Str -> (Number => Gender => Str) =
      \m,f,p -> table {
        Sg => genTbl m f ;
        Pl => \\_ => p
      } ;

  lincat Kind = { s : Number => Str ; g : Gender } ;
  lin
    Cheese = { s = numTbl "ġobna" "ġobniet" ; g = Fem };
    Fish   = { s = numTbl "ħuta"  "ħut"     ; g = Fem };

  lincat Quality = { s : Number => Gender => Str } ;
  lin
    Expensive = { s = numGenTbl "għali" "għalja" "għaljin" } ;
    Delicious = { s = numGenTbl "tajjeb"  "tajba"    "tajbin" } ;
    Very quality = { s = \\n,g => quality.s ! n ! g ++ ħ"afna"} ;

  lincat Item = { s : Str ; n : Number ; g : Gender } ;
  lin
```

```
    This kind = {
      s = (genTbl "dan il-" "din il-") ! kind.g ++ kind.s ! Sg ;
      n = Sg ; g = kind.g ;
      } ;
    These kind = {
      s = "dawn il-" ++ kind.s ! Pl ;
      n = Pl ; g = kind.g ;
      } ;

  lincat Comment = { s : Str } ;
  lin  Pred item quality = {
         s = item.s ++ copula ! item.n ! item.g ++ quality.s ! item.n ! item.g
         } ;
  oper copula = numGenTbl "huwa" "hija" "huma" ;
}
```

1.3.2 The GF runtime

GF includes a runtime system for working with grammars. It is available in various platforms: a shell application, libraries in Haskell, C and Java[10], and as a web service[11]. The runtime supports incremental parsing (Angelov, 2009), which allows it to parse incomplete input from the user and suggest the possible completions at the current point.

Example usage

Using the grammars developed in the previous section, we can show some examples of the use of GF by experimenting with parsing and linearisation in the GF shell. We first start the GF runtime and load the grammar files:

```
$ gf FoodsEng.gf FoodsMlt.gf
- compiling Foods.gf...   write file Foods.gfo
- compiling FoodsEng.gf...   write file FoodsEng.gfo
- compiling FoodsMlt.gf...   write file FoodsMlt.gfo
linking ... OK
Languages: FoodsEng FoodsMlt
Foods>
```

First, we can **parse** a comment from English into an abstract syntax tree (AST):

```
Foods> parse -lang=Eng "this fish is delicious"
Pred (This Fish) Delicious
```

We can also go in the opposite direction and **linearise** from an AST into some or all concrete languages:

[10] https://github.com/GrammaticalFramework/JPGF, accessed 2013-06-08
[11] https://code.google.com/p/grammatical-framework/wiki/GFWebServiceAPI, accessed 2013-06-08

```
Foods> linearize Pred (These Cheese) (Very Expensive)
these cheeses are very expensive
dawn il- ġobniet huma għaljin ħafna
```

Finally we can perform a **translation** from Maltese to English by piping the output of a parse into a linearise command:

```
Foods> parse -lang=Mlt "din il- ġobna hija tajba ħafna" | linearize -lang=Eng
this cheese is very delicious
```

1.3.3 Resource Grammar Library

The example grammar in the previous section is small, but as grammars grow in size having a module system becomes vital. Both abstract and concrete syntaxes can be split into separate modules, which promotes not only manageability but more importantly code re-use. Because the morphological and syntactic details of natural languages can be quite complex, it doesn't make sense for every new grammar written in GF to have to re-implement these features. Instead, the low-level details of a language can be put into a **resource grammar** whose abstract syntax models phenomena on the syntactic level. A separate **application grammar** can then be written which focuses on the high-level modelling tasks, using functions defined in the resource grammar as needed. This additional level of abstraction in grammar design means that code for semantic modelling does not need to be mixed with code that handles syntax and morphology.

Apart from the compiler and runtime system, the GF distribution also includes a standard library known as the Resource Grammar Library (RGL) (Ranta, 2009). The RGL provides general linguistic descriptions for natural languages which can be reused by using a common language independent API. This library covers the morphology and basic syntax of some 28 languages from around the world (where Maltese is the 28th language to be added). This means programmers who wish to use GF to write some application grammar do not need to worry about the tedious low-level details like word order and agreement in each language; they simply use the standard functions provided by the API and the linearisation into each language is handled automatically by GF.

Using the RGL in an application grammar

Continuing with the `Foods` example, we now show an example of the English concrete syntax which uses the Resource Grammar Library. We name this module `FoodsEngRGL`. Note how this module opens `SyntaxEng` and `ParadigmsEng`, which together make up the API to the library. In comparison to the previous version of `FoodsEng` above, `FoodsEngRGL` does not encode any linguistic information itself. No parameters are defined and no agreement handling takes place; Instead, the categories are given linearisation types directly from the RGL; for example, `Kind` is given the type of a common noun `CN`. The function linearisations also look simpler — instead

of manually constructing records, API constructors are used. So the linearisation of `Cheese` is defined as `mkCN (mkN "cheese")`, where the `mkN` smart paradigm will automatically generate plural forms of the supplied lemma `"cheese"`.

```
concrete FoodsEngRGL of Foods = open SyntaxEng, ParadigmsEng in {
  lincat
    Comment = Utt ;
    Item = NP ;
    Kind = CN ;
    Quality = AP ;
  lin
    Cheese = mkCN (mkN "cheese") ;
    Fish = mkCN (mkN "fish" "fish") ;
    Expensive = mkAP (mkA "expensive") ;
    Delicious = mkAP (mkA "delicious") ;
    Very quality = mkAP very_AdA quality ;
    This kind = mkNP this_Det kind ;
    These kind = mkNP these_Det kind ;
    Pred item quality = mkUtt (mkCl item quality) ;
}
```

1.3.4 Lexical resources

The RGL contains a small 300-word lexicon which is common to all languages in the library. This is mainly intended for testing purposes, and is far too limited for most real-world applications. Application grammars typically bear the task of defining the lexicons they require, but there has also recently been a focus on large-scale GF lexicons to complement the standard API.

Because of the work involved in lexicon compilation, large-scale GF lexicons are also created via extraction from existing sources, and can be either monolingual or multilingual. Monolingual lexicons in GF tend to be idiomatic, in that they are tailored for each language, and are intended as a resource or for a monolingual application. In other words, they do not provide any links between senses in different languages and thus cannot be used for translation tasks.

Multilingual lexicons on the other hand, share a common abstract syntax and can be used for multilingual applications. Experiments in extracting such resources have been carried out using the English WordNet and its translations into Bulgarian, German, Finnish and Hindi (Angelov *et al.* , 2013; Virk, 2013). A typical problem with this approach is the difficulty with exact word-sense matching. **Uni-sense** lexicons get around this problem by giving one-to-one correspondences between *lemmas* in the source and target. They are lightweight, simple, and can yield good results in many cases, but the word choice is often arbitrary. **Multi-sense** lexicons can be built from WordNet synsets, where every distinct word sense gets an entry in the lexicon. These can either include all combinations of lemma and word sense in abstract syntax, or use only one synonym per word sense. The former option is useful for parsing, but increases the size of the lexicon dramatically; the latter option is adequate for just linearisation

purposes.

1.4 Goals

This section summarises the motivations for this work and outlines its planned contributes.

1.4.1 Motivation

As neatly summarised in figure 1.1, Maltese is very poorly represented in digital terms. Despite there being an known academic interest in this area for at least 15 years, both the breadth and depth of linguistic tools and resources available for Maltese is still quite limited. The recently created Maltese corpus is by far the biggest milestone in the field to date, but there are many other areas of Maltese language technology which are lacking. As a native speaker of the Maltese, the motivation to contribute to this field is therefore clear.

1.4.2 Contributions

The goals main body of this work can be divided into two major contributions.

A computational grammar

The Resource Grammar Library has seen a steady growth in terms of language support, and to date covers 27 languages from all over the world. Complete RGL implementations exist for Bulgarian, Catalan, Chinese, Danish, Dutch, English, Finnish, French, German, Greek, Hindi, Interlingua, Japanese, Italian, Latvian, Nepali, Norwegian bokmål, Persian, Polish, Punjabi, Romanian, Russian, Sindhi, Spanish, Swedish, Thai, and Urdu. Partial implementations also exist for a number of other languages, and all resources are completely open-source under the LGPL license. Maltese will be the 28^{th} addition to the library, and the first Semitic language to have a complete implementation. A resource grammar implementation for Maltese will:

1. Contribute directly to the Grammatical Framework by extending the coverage of its standard library.
2. Provide a system for producing morphological inflection tables and performing morphological analysis in Maltese.
3. Act as a limited-domain parser and translator to and from all other languages available in the RGL, without requiring individual rules per language pair.
4. Enable software developers using to GF to provide interface localisations into Maltese without requiring manual translations.

A computational lexicon

As an initial step to the creation of a lexicon, a platform for the **collection of lexical resources** in Maltese will be created. This collection will be hosted online and searchable in a single way, and all data should be extractable and easily convertible into other formats. It will initially be populated with data from the resources listed in section 1.2.5.

Choosing a highly flexible storage representation will allow new resources to be easily added to the collection as they become available. In this way we hope to enable to organic growth of a computational lexicon for Maltese, and thus avoid some of the startup problems encountered in previous attempts at building one (see section 1.2.3).

By then combining this collection of lexical resources with the morphological generation from the resource grammar, we will extend this collection of resources into a **full-form lexicon**. Such lexicons are useful for spell checking and lemmatisation, particularly in morphologically-rich languages such as Maltese where the concept of automatically-derivable word stem is not so prominent. Including all inflectional forms in a lexicon will increase the number of word forms by a few orders of magnitude (in the case of Maltese). This work is very repetitive but also not viably done manually, making it an ideal candidate for rule-based production by a computational grammar. This full-form lexicon will take two forms:

1. A searchable online database which stores all word forms (both generated and manually imported from other sources).

2. A monolingual GF dictionary module (`DictMlt.gf`), which uses smart paradigms to keep the lexicon as compact as possible.

1.4.3 Organisation

The remainder of this monograph is organised as follows. Chapter 2 covers the implementation of the Maltese resource grammar, including the testing carried out during development. Chapter 3 then covers the design of the platform for the collection of lexical resources, using the resource grammar to generate full inflection forms and the creation of a monolingual dictionary module. Finally in chapter 4 we make some conclusions about the value of this thesis in the context of other related works and discuss some directions for future work.

The appendices contain a description of the RGL API and the lexical paradigms available in the Maltese implementation, analyses of certain linguistic phenomena relevant to the design of the resource grammar, and information about licensing and obtaining the source code for all the work described in this thesis.

Chapter 2

Resource grammar

This chapter details the implementation of a computational grammar for Maltese, using the Grammatical Framework. Various phenomena in Maltese morphology and syntax are highlighted and their implementation in the resource grammar explained. We then look at the techniques used for testing the resource grammar and discuss its coverage. The chapter finishes by looking at some of the implementational problems encountered during development.

2.1 Module structure

The Maltese resource grammar consists of 28 GF modules and follows the same module structure as most of the other language implementations in the library. A module dependency graph and descriptions of the contents of each module can be found in Appendix A. In line with convention, all Maltese modules in the library are suffixed with the ISO 639-3 letter code `Mlt`. All modules are located in GF's source code repository in the directory `lib/src/maltese`. More information about accessing the source code can be found in Appendix D.

A number of related languages in the RGL use a functor implementation, which allows them to share syntactic constructs which are common and only override what differs in each language. Examples include the *Romance* functor, which used in Catalan, French, Italian and Spanish; the *Scandinavian* functor for Danish, Norwegian (bokmål) and Swedish; and a *Hindustani* functor which powers the Hindi and Urdu resource grammars. The primary advantage of having a functor implementation is code re-use, however the process of designing a common functor can be quite involved.

The Maltese grammar does not use such an implementation. Despite containing elements of both Semitic and Romance morphology, there is simply not enough in common with other languages in the library to warrant the extra work involved in setting up a functor. Additionally, the two languages arguably most closely related to Maltese — Arabic and Hebrew — do not have complete RGL implementations.

2.2 Morphology

2.2.1 Smart paradigms

A morphological **paradigm** describes the complete set of word forms which are associated with a lexeme, such as the conjugations of a verb and the declensions on a noun. In GF, a paradigm translates to a function which takes one or more representative word forms of a lexeme and constructs a complete inflection table for it. For example calling the strong verb paradigm function with the lemma *waqaf* ('he stopped') like so:

```
strongV "waqaf"
```

gives the full inflection table below:

```
s (VPerf (AgP1 Sg)) : waqaft
s (VPerf (AgP1 Pl)) : waqafna
s (VPerf (AgP2 Sg)) : waqaft
s (VPerf (AgP2 Pl)) : waqaftu
s (VPerf (AgP3Sg Masc)) : waqaf
s (VPerf (AgP3Sg Fem)) : waqfet
s (VPerf AgP3Pl) : waqfu
s (VImpf (AgP1 Sg)) : nieqaf
s (VImpf (AgP1 Pl)) : nieqfu
s (VImpf (AgP2 Sg)) : tieqaf
s (VImpf (AgP2 Pl)) : tieqfu
s (VImpf (AgP3Sg Masc)) : jieqaf
s (VImpf (AgP3Sg Fem)) : tieqaf
s (VImpf AgP3Pl) : jieqfu
s (VImp Sg) : ieqaf
s (VImp Pl) : ieqfu
s (VPresPart (GSg Masc)) : wieqaf
s (VPresPart (GSg Fem)) : wieqfa
s (VPresPart GPl) : wieqfin
s (VPastPart (GSg Masc)) : mwaqqaf
s (VPastPart (GSg Fem)) : mwaqqfa
s (VPastPart GPl) : mwaqqfin
```

Natural languages typically have many paradigms which a lexeme may follow — often hundreds — and having to manually select the correct paradigm for each entry in a lexicon can be laborious. A **smart paradigm** (Détrez & Ranta, 2012) is a meta-paradigm which inspects the given base form of a lexeme and tries to infer which lower-level paradigm applies. In cases where the correct paradigm simply cannot be determined, more word forms are given for discrimination. In general, the average number of forms needed is a measure of the predictability of the morphology in a language. The paradigms made available as part of the Maltese resource grammar are listed in appendix A.

2.2.2 Verb

The verb is arguably the most interesting aspect of Maltese morphology. As described in section 1.1.2, Maltese verbs can be divided more or less cleanly in two groups: those which follow a Semitic root-and-pattern morphology, and those which display a concatenative morphology. (Spagnol, 2011) Most of the work on morphology concerns the former group as their conjugations are much more involved.

Etymologically, verbs in Maltese can roughly be divided into Semitic, Romance and English origin. While all the Semitic verbs follow a root-and-pattern morphology, there is a division in the Romance class of verbs. Some have become **strongly-integrated** into the root and pattern system and behave as quadriliteral verbs. Examples include *kanta* ('he sang'), *vinċa* ('he won') and *serva* ('he served'), from the Italian *cantare*, *vincere* and *servire* respectively. These are treated as Semitic verbs as quadriliteral verbs with the weak *j* as final radical.

The remainder of Maltese verbs from Romance origin are only **loosely-integrated** and follow a concatenative morphology of stems and affixes. Examples include *żviluppa* ('he developed'), *ipperfezzjona* ('he perfected'), *antagonizza* ('he antagonised'). The third group of verbs have an English origin, and follow the same morphology as the loosely-integrated Romance verbs, Examples include *ibbukja* ('he booked'), *ipparkja* ('he parked'), *iffittja* ('he fitted'). Despite their different etymology, these two latter groups behave identically from a morphological perspective and thus are not distinguished between in the grammar; they are both treated generically as **loan** verbs.

Classification

The classification of Maltese verbs is covered significantly elsewhere (Borg & Azzopardi-Alexander, 1997; Brother Henry F.S.C., 1980; Spagnol, 2011; Fabri, 2009). While broadly following this linguistic tradition, we are also interested in keeping representations and compact efficient. Table 2.1 shows the verb classification as used in the resource grammar. The **liquid-medial** class corresponds to Fabri's ITb paradigm (Fabri, 2009) (where **regular** is ITa), which require an epenthetic vowel to be inserted in the imperfective paradigm (Spagnol, 2011, p. 38). As noted but Fabri (2009), *għ* is not weak, thus defective verbs are technically strong. However, they behave inflectionally as weak verbs and are thus classified as such in our case.

Table 2.1: Verb classes and their characteristics

Class	Description	Example
Strong		
Regular	All radicals strong & distinct	*qatel* ('he killed') ($\sqrt{\text{QTL}}$)
Liquid-medial	C2 is liquid	*żelaq* ('he slipped') ($\sqrt{\text{ŻLQ}}$)
Geminated	C2 & C3 are identical	*ħabb* ('he loved') ($\sqrt{\text{ĦBB}}$)
Weak		

Continued on next page

Assimilative	C1 is weak	*waqaf* ('he stopped') (√WQF)
Hollow	C2 is weak; long vowel between C1 & C3	*dam* ('he delayed') (√DWM)
Lacking	C3 is weak	*beka* ('he cried') (√BKJ)
Defective	C3 is silent *għ*	*qata'* ('he cut') (√QTGĦ)
Quadriliteral		
Strong	All radicals strong	*ħarbat* ('he disrupted') (√ĦRBT)
Weak-final	C4 is weak	*pinġa* ('he drew') (√PNĠJ)
Irregular	Missing radicals or other irregularities	*ħa* ('he took') (√'ĦD)
Loan	Concatenative morphology	*pparkja* ('he parked')

Root-and-pattern verbs

Table 2.2 shows the inflection table for the Semitic verb *ħareġ* ('he went out'). Note how there is no stem invariant throughout the table; instead the radicals √ĦRĠ always occur in the same order, though the vowels move around and even change altogether.

Table 2.2: Inflection table for Semitic verb *ħareġ* ('he went out')

Subject	Perfective	Imperfective	Imperative
P1 Sg	*ħriġt*	*noħroġ*	–
P2 Sg	*ħriġt*	*toħroġ*	*oħroġ*
P3 Sg Masc	*ħareġ*	*joħroġ*	–
P3 Sg Fem	*ħarġet*	*toħroġ*	–
P1 Pl	*ħriġna*	*noħorġu*	–
P2 Pl	*ħriġtu*	*toħorġu*	*oħorġu*
P3 Pl	*ħarġu*	*joħorġu*	–

Derived verbs

Root-and-pattern verbs in Maltese also exhibits derivational morphology. Derived verbs generally indicate transitivity and/or passiveness with respect to a first-form verb. The different derivational processes are well understood but they occur sporadically; that is to say, no verb root has instances of all derived forms. The list of such verbs is closed in Maltese and is no longer productive. These have been exhaustively enumerated by Spagnol (2011).

In the resource grammar, derived verbs are treated simply as separate lexemes in the lexicon. A single parameter in the `Verb` type indicates what derived form a verb is in, which determines its morphological paradigm. Tables 2.3 and 2.4 give examples of each of the derived forms for both tri- and quadriliteral verbs.

Table 2.3: Derivational forms for triliteral verb roots

Form	Process	Example	Description
I	–	*kiteb* ('he wrote')	Base form
II	Reduplicate C2	*daħħal* ('he brought in')	Intransitive I verbs are made transitive / causative.
III	Lengthen v1	*bierek* ('he blessed')	As in II, when reduplication is not possible.
IV	–	*wera* ('he showed')	(Empty class)
V	Prefix *t-* to II	*tfarrak* ('it was shattered')	Intransitive. Passive or reflexive.
VI	Prefix of *t-* to III	*tbierek* ('he was blessed')	Same as V.
VII	Prefix of *n-* to I	*nkiteb* ('it was written')	Transitive I verbs are made intransitive.
VIII	Infix *-t-* after C1 of I	*ftiehem* ('he understood')	Intransitive (like V, VI, VII).
″	Prefix *nt-* to I	*ntlaħaq* ('it was reached')	″
″	Infix *n-t-* to I	*nxteħet* ('he sprawled out')	″
IX	Structure CCv:C	*ċkien* ('it reduced in size')	Change of state.
X	Prefix *st-*	*stagħġeb* ('he was astonished')	–

Table 2.4: Derivational forms for quadriliteral verb roots

Form	Process	Example	Description
I	–	*ħarbat* ('he disrupted')	Base form
II	Prefix *t-*	*tħarbat* ('he was disrupted')	Intransitive. Passive or reflexive.

Concatenative verbs

Loosely-integrated Romance verbs and English loan verbs in Maltese follow an alternative, concatenative morphology. Table 2.5 shows the inflection table for the English-loan verb *pparkja* ('he parked'). Note how in contrast to the root-and-pattern verb inflection (table 2.2), the stem *pparkja* remains intact throughout the entire table.

Table 2.5: Inflection table for an English-loan verb *pparkja* ('he parked')

Subject	Perfective	Imperfective	Imperative
P1 Sg	*pparkjajt*	*nipparkja*	–
P2 Sg	*pparkjajt*	*tipparkja*	*pparkja*
P3 Sg Masc	*pparkja*	*jipparkja*	–

Continued on next page

P3 Sg Fem	*pparkjat*	*tipparkja*	–
P1 Pl	*pparkjajna*	*nipparkjaw*	–
P2 Pl	*pparkjajtu*	*tipparkjaw*	*pparkjaw*
P3 Pl	*pparkjaw*	*jipparkjaw*	–

Mifsud (1995) identifies four classes of loan verbs (types A to D), of which types C and D are not integrated into the Semitic root-and-pattern system. These represent the non-integrated verbs of Romance origin (e.g. *ppretenda* 'he expected') and English loan verbs (e.g. *iffitja* 'he fitted'). Despite Mifsud's distinction, we treat these two classes of loan verbs with the same paradigm in the resource grammar.

Participles

Maltese verbs may also have participle forms. Most verbs have a **past/passive** participle which is often indistinguishable from the adjective. Examples include *miktub* ('written') from *kiteb* ('he wrote') and *misruq* ('stolen') from *seraq* ('he stole'). Very few verbs also have a **present/active** participle; they are generally intransitive and describe actions, for example *ħiereġ* ('going out') from *ħareġ* ('he went out') and *rieqed* ('sleeping') from the verb *raqad* ('he slept').

A part participle may be shared by verbs of different derived formed, such as *mwaqqaf* ('stopped') for both *waqaf* ('he stopped') (form I) and *waqqaf* ('he stopped [s.t.]') (form II). An analysis of participles of all the verbs in the `LexiconMlt` module can be found in appendix B.2.

Enclitic pronouns

Maltese verbs can take suffixed enclitic pronouns to indicate direct objects, indirect objects, and both combined together. Table 2.6 displays how this works. Refer to appendix C for an exhaustive example of these combinations. The implementation of enclitic pronouns in the grammar is explained further in section 2.3.3.

Table 2.6: Different combinations of suffixed enclitic pronouns to the verb *xtara* ('he bought')

D.O.	I.O.	Construction	Final form	English gloss
-	-	*xtara* + Ø	*xtara*	'he bought'
P3 Sg Fem	-	*xtara* + *hija*	*xtaraha*	'he bought it/her'
-	P3 Pl	*xtara* + *lilhom*	*xtaralhom*	'he bought for them'
P3 Sg Fem	P3 Pl	*xtara* + *hija* + *lilhom*	*xtarahielhom*	'he bought it/her for them'

Implementation

Record type The record type for the verb in the resource grammar is defined as follows:

```
oper
```

```
  Verb : Type = {
    s : VForm => VerbStems ;
    i : VerbInfo ;
    hasPresPart : Bool ;
    hasPastPart : Bool ;
    } ;
  VerbStems : Type = {s1, s2, s3 : Str} ;
param
  VForm = VPerf VAgr | VImpf VAgr | VImp Number | VActivePart GenNum | VPassivePart GenNum ;
  VAgr  = AgP1 Number | AgP2 Number | AgP3Sg Gender | AgP3Pl ;
```

where `VerbInfo` is a record type containing information about verb class, radicals and derived form number.

The `VForm` parameter defines the different inflection forms that a verb may have, covering the perfective and imperfective aspects and the imperative mood with corresponding subject agreement features. In addition to these, we also include in the verb table the present and past participle forms of the verb, which is common practice in the RGL. Since not all verbs have these participles, we also include the two boolean fields `hasPresPart` and `hasPastPart` to indicate their presence.

Rather than having a table from `VForm` into a string, we instead use the record of strings `VerbStems` to contain the different stems that may be needed when the base verb form has enclitic pronouns attached to it. For more information about this, refer to section 2.3.3.

Paradigms `mkV` is the smart paradigm for all non-derived verbs. This operator minimally takes the *mamma* form of a verb and a root constructed with `mkRoot` in the case of root-and-pattern verbs. The root class is determined and the correct paradigm is applied. If no root is specified as an argument, the verb is treated as a loan verb (with concatenative morphology). Vowel changes are often hard to determine, thus the singular imperative form of a verb is often required as a third argument. For irregular verbs, we can take all 16 forms (7 perfective, 7 imperfective, 2 imperative) together with class, derived form, root and vowel sequence. Verb participles are specified not in the `mkV` operator but by using the supplementary operators `presPartV` and `pastPartV` which add participles to already-constructed verbs. Derived verbs have their own set of paradigms, as they are treated as separate entries in the lexicon. Examples are given below:

Explanation	**Constructor**
Geminate root-and-pattern verb	`mkV "ħabb" (mkRoot "ħ-b-b")`
Requiring an extra form	`mkV "talab" "itlob" (mkRoot "t-l-b")`
Loan verb	`mkV "pparkja"`
Form II (derived) verb	`mkV_II "daħħal" (mkRoot "d-ħ-l")`

2.2.3 Noun

The noun in Maltese has an inherent gender and inflects for number. The number is not simply singular/plural though, as Maltese nouns may have collective and dual forms. The collective is syntactically singular but semantically plural, e.g. *dan it-tuffieħ ħelu* ('these apples are sweet') (as opposed to *dawn it-tuffieħ ħelwin*. Some Maltese nouns also have a dual, typically parts of the body and measures of time. Nouns can have two types of plural: determinate and indeterminate. Table 2.7 summarises these different forms.

Table 2.7: Noun numbers

Form	Count	Example
Singular	1, more than 10	*ħuta*
Collective	Not numerically quantifiable	*ħut*
Dual (*għadd imtenni*)	2	*ġimgħatejn*
Determinate plural	2-10	*triqat*
Indeterminate plural	Not numerically quantifiable	*toroq*

Nouns typically have some subset of these forms, though never all of them together. The linguistic literature does not identify and paradigms for nouns in this sense, thus some further investigation was needed. By surveying the 186 nouns in the RGL mini-lexicon (`Lexicon.gf`), four different noun paradigms have been identified. There are summarised in table 2.8. Refer to see appendix B.1 for the full analysis.

Table 2.8: Noun paradigms

Paradigm	Singular	Collective	Dual	Det. Plural	Ind. Plural
1	*qattus*	-	-	*qtates*	-
1x	*triq*	-	-	*triqat*	*toroq*
2	*nagħġa*	*ngħaġ*	-	*nagħġiet*	-
2x	*frotta*	*frott*	-	*frottiet*	*frottijiet*
2b	-	*persuna*	-	*persuni*	-
2bx	-	*ħalib*	-	*ħalibijiet*	*ħlejjeb*
2c	-	*ċpar*	-		-
3	*pulizija*	-	-	-	-
4	*rkoppa*	-	*rkopptejn*	*rkoppiet*	
4x	*għajn*	-	*għajnejn*	*għajnejn*	*għejun*

The analysis also reveals that very few nouns actually have both a determinate and indeterminate plural form, and when they do one often sounds arcane. While this distinction can have some linguistic importance, this is simplified in the GF implementation by storing only

one plural form. This change will be made internally in the noun representation, so that the paradigm constructors can still take both forms and thus changed at a later point if needed. Another solution would be to have indeterminate plural forms stored as variants of the determinate plural.

Plural formation

The noun plural can be formed by various morphological processes, as showed in table 2.9.

Table 2.9: Morphological processes for plurals

Type	Process	Singular	Plural
Sound	Suffixation (external)	*fergħa*	*fergħat*
Broken	Internal change	*fergħa*	*friegħi*
Plural of plural	Broken plural + suffixation	*tarf*	*trufijiet* (from *truf*)
Irregular	Entirely different	*mara*	*nisa*
Foreign (Romance)	Suffix change $a \rightarrow i$	*karta*	*karti*
Foreign (English)	Suffixation *+s*	*televixin*	*televixins*

Enclitic pronouns

Some nouns in Maltese can take enclitic pronouns to mark possession. For example, the word *id* ('hand') takes the suffixed for of the pronoun *tiegħi* ('mine') to form *idi* ('my hand'). The group of nouns that allow this is relatively small and consists mostly of body parts, but the process is certainly common enough to be handled in the grammar. The implementational details of enclitic pronouns are discussed in section 2.3.3.

Implementation

Record type The record type for nouns is as follows:

```
oper
  Noun : Type = {
    s : Noun_Number => Str ;
    g : Gender ;
    hasColl : Bool ;
    hasDual : Bool ;
    takesPron : Bool ;
    } ;
param
  Noun_Number = Singulative | Collective | Dual | Plural ;
```

Boolean fields `hasColl` and `hasDual` are required since GF does not support checking string values at runtime (see section 2.5.4). `takesPron` specifies whether a noun takes enclitic pronouns

as described above. This is specified manually in the paradigm as there is no way it can be determined automatically.

Proper nouns are much simpler as they do not inflect for any feature:

```
oper
  ProperNoun : Type = {
    s : Str ;
    a : Agr ;
    } ;
```

Paradigms Noun construction operators exist for each of the paradigms listed in table 2.8. Examples of each are given in table 2.8.

Table 2.10: Noun paradigms

No.	Explanation	Constructor
1	Smart paradigm	`mkN "rota"`
1	Specifying broken plural	`mkN "għasfur" "għasafar"`
1x	Both plural forms and explicit gender	`mkN "triq" "triqat" "toroq" feminine`
2	Singular, collective and plural forms	`mkNColl "nagħġa" "ngħaġ" "nagħġiet"`
2b	Collective and plural form	`mkNColl "persuna" "persuni"`
2c	Only collective form	`mkNColl "ċpar"`
3	Only singular form	`mkNNoPlural "plastik"`
4	Noun with dual form	`mkNDual "sena" "sentejn" "snin"`

Similarly to verbs, nouns in Maltese of Semitic origin are associated with a consonantal root. For example, the root $\sqrt{\text{XRB}}$ yields both the verb *xorob* ('he drank') as well as the noun *xorb* ('[alcoholic] drink'). However the root-and-pattern system is much less predictable in the case of nouns, and no consistent patterns have been found for constructing broken plurals. Thus any broken plurals must be specified explicitly, and there are no special paradigms for nouns with Semitic roots. The only 'smart' behaviour in the noun paradigms are simple vowel changes when the singular form ends in an 'a', e.g. *rota* → *roti*.

2.2.4 Adjective

By comparison to nouns and verbs, the adjective in Maltese is very straightforward. It inflects for number (singular or plural), and gender in the singular. When an adjective has an inflectional comparative, it does not inflect at all. Otherwise, the comparative is formed syntactically with the structural words *iktar* or *inqas*, "more" and "less" respectively. In this case the correctly inflected positive form of the adjective needs to be selected. The superlative is always formed syntactically by adding the definite article to the comparative. Table 2.11 gives examples of these forms.

Table 2.11: Inflection table for two adjectives *sabiħ* ('beautiful') and *kiesaħ* ('cold'), respectively with and without inflectional comparative forms

Agreement	Positive	Comparative	Superlative
Sg Masc	*sabiħ*	*isbaħ*	*l-isbaħ*
Sg Fem	*sabiħa*	″	″
Pl	*sbieħ*	″	″
Sg Masc	*kiesaħ*	*iktar kiesaħ*	*l-iktar kiesaħ*
Sg Fem	*kieshа*	*iktar kieshа*	*l-iktar kieshа*
Pl	*keshin*	*iktar keshin*	*l-iktar keshin*

Implementation

Record type The record type looks like this:

```
oper
  Adjective : Type = {
    s : AForm => Str ;
    hasComp : Bool ;
    } ;
param
  AForm = APosit GenNum | ACompar | ASuperl ;
```

Paradigms The smart paradigm for constructing adjectives can take between 1 and 4 strings, depending on the predictability of the inflected forms and whether it has a comparative form or not. A second paradigm `sameA` exists for adjectives which are invariant (and have no comparative form). Examples of all are given in table 2.12.

Table 2.12: Adjective paradigms

Explanation	Constructor
Predictable feminine and plural forms	`mkA "bravu"`
Predictable feminine, explicit plural form	`mkA "nadif" "nodfa"`
Explicit feminine and plural forms	`mkA "aħmar" "ħamra" "ħomor"`
All forms, including comparative	`mkA "dejjaq" "dejqa" "dojoq" "idjaq"`
Invariant inflection	`sameA "blu"`

2.2.5 Other lexical categories

In addition to the simple lexical categories given above, the Resource Grammar Library also defines higher-valency versions of these categories. The explanation of the `V` category for ex-

ample is a simple one-place verb. Entries from the lexicon with this type include come, fly and sleep. Two-place verbs `V2` are ones which require an object, such as hate, love, and read. They can also be thought of as transitive verbs. The `V3` category similarly encodes three-place or ditransitive verbs. Higher-valency categories also exist for nouns (`N2`, `N3`) and adjectives (`A2`).

The linearisation types for these categories typically resemble their basic versions, with some extra field. Thus `V2` is defined as a verb with an additional complement field which contains a preposition:

```
lincat
  V2 = Verb ** {c2 : Compl} ;
oper
  Compl : Type = Preposition ** {isPresent : Bool} ;
```

The linearisation of such a two-place verb combines the standard `mkV` paradigm with an additional preposition:

```
lin
  love_V2 = prepV2 (mkV ħ"abb" (mkRoot ħ"-b-b")) (mkPrep "lil") ;
oper
  prepV2 v p = lin V2 ( v ** { c2 = hasCompl p } ) ;
```

2.2.6 Euphonic vowels

There are many cases when euphonic vowels (*vokali tal-leħen*) are inserted in Maltese orthography to match the phonetic pronunciation. This typically takes the form of the letter *i* which is prefixed to a word beginning with a double consonant. It would however seem that there is some variation as to whether this vowel is considered part of the base form or not. Using the Serracino-Inglott (2003) dictionary, we see that sometimes the *i* is part of the lemma form, and sometimes it isn't. Table 2.13 gives some examples of this variation.

In the case of root-and-pattern verbs, it seems to be a matter of convention which is applied throughout. While (Borg & Azzopardi-Alexander, 1997) gives verb conjugations **with** initial vowels: *insum, isum, inbierek, ibierek* etc., others (Serracino-Inglott, 2003; Azzopardi, 2007; Brother Henry F.S.C., 1980) prefer to leave them out and give the standard form without the euphonic vowels: *nsum, jsum, nbierek, jbierek* etc.

Table 2.13: Variation in inclusion of euphonic *i* in lemma forms, as based on the Maltese dictionary by Serracino-Inglott (2003)

Class	Examples without	Examples with
Noun	*mbuljuta, mnieħer, nbid, spettur*	*imbatt, impenn, injam, istitut*
Loan verb	*mmatura, kkopja, pparkja, nnervja*	*immaġina, ikkoppja, ipponta, innamra*
Adjective	*mħassar, mxarrab, rqiq*	*importanti, inċert, isfar*

Table 2.14 shows a few words from different lexical categories which differ in terms of whether their base form includes a vowel or not. What is clear from this small comparison is that when the euphonic *i* vowel is part of the base form, then it is never removed. The rules for euphonic vowels are covered extensively in (Azzopardi, 2007, ch. 10).

Table 2.14: Behaviour of words with/without initial euphonic vowels when combined with other words and particles

Noun	**Preceding vowel**	**Preceding consonant**	**Article**	**Preposition**
nbid	*ħafna nbid*	*ftit nbid*	*l-inbid*	*b'inbid*
mqarrun	*ħafna mqarrun*	*ftit mqarrun*	*l-imqarrun*	*b'imqarrun*
mħabba	*ħafna mħabba*	*ftit imħabba*	*l-imħabba*	*b'imħabba*
skola	*ħafna skejjel*	*ftit skejjel*	*l-iskola*	*bi skola*
injam	*ħafna injam*	*ftit injam*	*l-injam*	*b'injam*
ilma	*ħafna ilma*	*ftit ilma*	*l-ilma*	*b'ilma*
nemla	*ħafna nemel*	*ftit nemel*	*in-nemla*	*b'nemla*
serp	*ħafna sriep*	*ftit sriep*	*is-serp*	*b'serp*
Verb				
nħobb	*jiena nħobb*	*jien inħobb*	–	–
mmur	*jiena mmur*	*jien immur*	–	–
mxejt	*jiena mxejt*	*jien imxejt*	–	–
jmur	*huwa jmur*	*kien imur*	–	–
bkejt	*jiena bkejt*	*jien bkejt*	–	–
Adjective				
isfar	*ilma isfar*	*injam isfar*	*l-isfar*	–
rqiq	*mara rqiqa*	*raġel irqiq*	*l-irqiq*	–

Another point of interest is the conversion which sometimes occurs between *j* and *i*. This is not fully handled in the resource grammar as the rule is not really strictly adhered to, and thus considered low in terms of importance.

2.3 Syntax

2.3.1 Particles

Article

Absence of an article in Maltese indicates the indefinite, e.g. *rajt baqra* ('I saw a cow'). The standard definite article is *il-* which is glued to the succeeding word, e.g. *rajt il-baqra* ('I saw the cow'). When that word begins with one of the coronal consonants, it is assimilated: *il-* + *dar* becomes *id-dar* ('the house'). If the word begins with a vowel, the *i* from the article is dropped:

l-ajruplan (‘the airplane’). Similarly if a word begins with a consonant pair including a liquid, a euphonic *i* is inserted and the vowel form of the article is used, e.g. *l-iskola* (‘the school’). Articles also join into some preceding prepositions, as described below.

Implementation The implementation of articles in the resource grammar uses the `pre` construct, which is specifically designed for modelling variations of a word based on the succeeding letter. The definition of the articles is shown below (the actual implementation uses an intermediary operator, which is equivalent to the following code):

```
oper
  artIndef : Str = [] ;
  artDef : Str = pre {
    -- Regular consonant
    "il-" ++ BIND ;
    -- Vowel
    "l-" ++ BIND / strs { "a" ; "e" ; "i" ; "o" ; "u" ; "h" ; "għ" } ;
    -- Coronal consonants
    "iċ-" ++ BIND / strs { "ċ" } ;
    "id-" ++ BIND / strs { "d" } ;
    "in-" ++ BIND / strs { "n" } ;
    "ir-" ++ BIND / strs { "r" } ;
    "is-" ++ BIND / strs { "s" } ;
    "it-" ++ BIND / strs { "t" } ;
    "ix-" ++ BIND / strs { "x" } ;
    "iż-" ++ BIND / strs { "ż" } ;
    "iz-" ++ BIND / strs { "z" }
  } ;
```

Preposition

Prepositions are very similar to those in the English language. When using a preposition with a noun phrase in the definite case, the article will always assimilate with the article, as shown in table 2.15.

Table 2.15: Assimilation of particles

Preposition	Article	Succeeding word	Result
–	*il-* (‘the’)	*bieb*	*il-bieb* (‘the door’)
–	*il-*	*dar*	*id-dar* (‘the house’)
–	*il-*	*skola*	*l-iskola* (‘the school’)
bi (‘with’)	–	*bieb*	*b’bieb* (‘with a door’)
bi	–	*dar*	*b’dar* (‘with a house’)
bi	–	*skola*	*bi skola* (‘with a school’)

Continued on next page

fi (‘in’)	*il-*	*bieb*	*fil-bieb* (‘in the door’)
fi	*il-*	*dar*	*fid-dar* (‘in the house’)
fi	*il-*	*skola*	*fl-iskola* (‘in the school’)

2.3.2 Numerals

While there did exist a GF implementation of numerals in Maltese[1], this was hugely simplified and quite incorrect. The `NumeralsMlt` module described here has been completely rewritten.

Cardinals

These have one of two forms depending on whether they appear in isolation or in a construct state. Azzopardi (2007, p. 133) calls these “type A” and “type B” numerals, which in (Brother Henry F.S.C., 1980, p. 202) are described as *użu aġġettiv* (‘adjectival use’) and *użu nominali* (‘nominal use’) respectively.

For singular nouns, the numeral for ‘one’ comes after the noun or omitted completely, e.g. *raġel [wieħed]* (‘one man’). In expressions of quantity from two to ten, the numeral precedes the noun which appears in its plural form: *erbgħa ħwienet* (‘four shops’). When expressing exactly two of a noun which has a dual form, the numeral is omitted and this form is used: *sentejn* (‘two years’). Beyond the number 10, the noun always appears in the singular, e.g. *tlieta u għoxrin tifla* (‘twenty-one girls’). Between eleven and nineteen, the particle *-il* (not to be confused with the definite article *il-*) is suffixed added to the numeral, e.g. *tnax-il tifel* (‘twelve boys’).

Hundreds and thousands are constructed in a similar way to English using *mija* and *elf* respectively, e.g. *tlett mija u erba’* (‘three hundred and four’), *elf, erba’ mija u għoxrin* (‘one thousand, four hundred and twenty’).

Ordinals

The ordinal numerals from one to ten have their own specific forms, which all include the definite article, e.g. *it-tielet premju* (‘the third prize’). Ordinals above the tenth are expressed by means of definite article with the respective cardinal form: *il-ħamsa u għoxrin ta’ Diċembru* (‘the twenty-fifth of December’). (Borg, 1974)

Implementation

The RGL supports numerals both in words and as digits, with the `Numeral` and `Digits` categories respectively. Their linearisation types are given below:

```
lincat
  Numeral = {
    s : CardOrd => NumCase => Str ;
    n : NumForm
```

[1] Available at http://www.grammaticalframework.org/examples/numerals/maltese.gf, accessed 2013-05-08

```
  } ;
  Digits = {
    s : NumCase => Str ;
    n : NumForm ;
    tail : DTail ;
  } ;
param
  CardOrd = NCard | NOrd ;
  NumForm = NumX Number | Num0 | Num1 | Num2 | Num3_10 | Num11_19 | Num20_99 ;
  NumCase = NumNom | NumAdj ;
```

Besides storing cardinal and ordinal forms for both adjectival and nominal use, these types also contain a `NumType` field. This is used when constructing noun phrases to determine the number form which should be chosen. This is done by the `numform2nounnum` operator:

```
oper
  numform2nounnum : NumForm -> Noun_Number = \n ->
    case n of {
      NumX Sg  => Singulative ;
      NumX Pl  => Plural ;
      Num0     => Singulative ;
      Num1     => Singulative ;
      Num2     => Dual ;
      Num3_10  => Collective ;
      Num11_19 => Singulative ;
      Num20_99 => Plural
    } ;
```

The implementation for `Digits` is not specific to Maltese, since these work as in English. There are no ordinal forms for digits in Maltese.

Binding The design of GF requires that all possible tokens are known at runtime. This makes it impossible to perform true suffixation to tokens in syntax-level rules. The recommended solution to this is using the special bind token `&+` between two tokens which should appear as a single token in linearisation.

For example, the noun *missier* ('father') may be suffixed with the P1 Sg enclitic pronoun *-i* (indicating possession) by producing a list of three tokens: `["missier", "&+", "i"]`. When passed trough a suitable unlexer, this is displayed as a single string `"missieri"`.

This solution is time and space efficient and works reasonably well for generation. However it introduces a new problem when it comes to parsing. Now the grammar cannot parse the complete form `"missieri"` as a single token; instead this must be lexed in a pre-process to GF to re-introduce the binding token. The problems with this are discussed further in section 2.5.2.

Implementation The linearisation type for pronouns is quite simple:

```
oper
  Pronoun = {
    s : PronForm => Str ;
    a : Agr ;
    } ;
param
  PronForm = Personal | Possessive | Suffixed ;
```

The `Suffixed` form of the pronoun is used when attaching to nouns and prepositions. The pronoun suffixes for verbs however are not stored as part of the `Pronoun` record. Rather, the agreement feature `a` in the pronoun is consulted directly and the correct suffixed form is chosen within the linearisation function of a verb phrase, using the `dirObjSuffix` and `indObjSuffix` operators in `ResMlt.gf`. For more on this see section 2.3.5.

2.3.3 Pronouns

The pronouns available in Maltese are very similar to those in English. Pronouns in Maltese are often enclitic, joining with verbs to indicate direct/indirect objects, with nouns to indicate possession and with pronouns as contractions. The suffix forms which the personal pronouns can take are listed in table 2.16. While these can be seen of as a morphological process[2], the implementation in the resource grammar treat this is a syntactic processes via the **bind** token[3].

Table 2.16: Different suffixed forms for enclitic pronouns. Square brackets indicate forms which precede another enclitic pronoun in the dative.

Subject	**Nominative**	**Noun/ Preposition**	**Verb accusative (direct)**	**Verb dative (indirect)**
P1 Sg	*jien*	*-i, -ja*	*-ni*	*-li*
P2 Sg	*int*	*-ek*	*-ek, -k*	*-lek*
P3 Sg Masc	*hu*	*-u, -h*	*-u, -h* [*-hu*]	*-lu*
P3 Sg Fem	*hi*	*ha*	*-ha* [*-hie*]	*-lha*
P1 Pl	*aħna*	*na*	*-na*	*-lna*
P2 Pl	*intom*	*kom*	*-kom*	*-lkom*
P3 Pl	*huma*	*hom*	*-hom*	*-lhom*

[2] Borg & Azzopardi-Alexander (1997) present pronominal suffixes as inflectional forms.

[3] An early version of the grammar treated all pronominal suffixes as part of the inflection table. In the case of verbs, this led to a table of around 1000 unique word forms which proved problematic to compile and use. Refer to section 2.5.1.

Pro-drop

Maltese is a pro-drop language, meaning that personal pronouns in the subject position are often dropped since the verb inflection is enough to indicate the subject. However it is also valid to include subject pronouns, thus the dropping of pronouns must be specified explicitly using the `ProDrop` function defined in the `Extra` module:

```
lin
  ProDrop p = {
    s = table {
      R.Personal => [] ;
      c => p.s ! c
      } ;
    a = p.a ;
    } ;
```

2.3.4 Tense-anteriority system

Maltese verbs inflect for mood (indicative and imperative), and in case of the former for aspect (perfective and imperfective). Tenses are formed by combining these with auxiliary verbs *kien* ('he was'), *ikun* ('he would be'), future marker *se/ser/sa* ('he will') and progressive marker *qed* ('he is'). Table 2.17 lists the tenses traditionally identified in Maltese linguistics. (Ebert, 2000; Fabri, 1995; Brother Henry F.S.C., 1980).

Table 2.17: Maltese tenses as described by Brother Henry F.S.C. (1980)

Aux 1	Aux 2	Main verb	Meaning	Gloss
–	–	*sraqt*	Simple past	'I stole'
kont	–	*sraqt*	Past perfect	'I had stolen'
nkun	–	*sraqt*	Future perfect	'I will have stolen'
–	–	*nisraq*	Habitual present	'I steal'
kont	–	*nisraq*	Habitual past	'I used to steal'
–	*qed*	*nisraq*	Present progressive	'I am stealing'
kont	*qed*	*nisraq*	Past progressive	'I was stealing'
nkun	[*qed*]	*nisraq*	Future progressive	'I will be stealing'
–	*se*	*nisraq*	Prospective	'I am going to steal'
kont	*se*	*nisraq*	Past prospective	'I was going to steal'
nkun	*se*	*nisraq*	Future prospective	'I will be going to steal'

The tense system in the GF Resource Grammar Library uses a combination of anteriority (simultaneous, anterior), temporal order (present, past, future, conditional) and polarity (positive, negative). This yields a total of 16 distinct temporal forms. Table 2.18 list these tenses and

shows how each of them is expressed in Maltese[4].

Table 2.18: RGL tense-anteriority system and correspondences to Maltese tenses

Anteriority	Tense	Polarity	Description	Example
Simult.	Present	Pos.	Imperfective	*jorqod*
Simult.	Present	Neg.	″	*ma jorqodx*
Simult.	Past	Pos.	Perfective	*raqad*
Simult.	Past	Neg.	″	*ma raqadx*
Simult.	Future	Pos.	Prospective	*se jorqod*
Simult.	Future	Neg.	″	*m'hux se jorqod*
Simult.	Conditional	Pos.	Past imperfective	*kien jorqod*
Simult.	Conditional	Neg.	″	*ma kienx jorqod*
Anterior	Present	Pos.	(as Sim Past Pos)	
Anterior	Present	Neg.	(as Sim Past Neg)	
Anterior	Past	Pos.	Past perfect	*kien raqad*
Anterior	Past	Neg.	″	*ma kienx raqad*
Anterior	Future	Pos.	Future perfect	*se jkun raqad*
Anterior	Future	Neg.	″	*m'hux se jkun raqad*
Anterior	Conditional	Pos.	Past prospective	*kien jorqod*
Anterior	Conditional	Neg.		*ma kienx jorqod*

2.3.5 Phrases

Noun phrase

As shown in figure A.1, a noun phrase combines all kinds of determiners, pronouns, proper and common nouns, and relative clauses. Most of these elements are linearised into strings in the formation of the noun phrase, such that the main s field is a record from case to string:

```
oper
  NounPhrase : Type = {
    s : NPCase => Str ;
    a : Agr ;
    isPron : Bool ;
    isDefn : Bool ;
    } ;
param
  NPCase = NPNom | NPAcc | NPCPrep ;
```

[4]This table differs from that shown in (Zammit, 2012, p. 31), which contains some errors.

The noun phrase also carries information about its agreement features which is needed when combining with verb phrases, as well as whether it is a pronoun and/or in the definite form. The `CPrep` constructor of the `NPCase` parameter is used to handle cases where the noun phrase is preceded by a preposition. In such cases the preposition may join with the definite article in the NP, or completely assimilate with the phrase if it is a pronoun. This implementation builds on that developed in (Zammit, 2012).

Verb phrase

The verb phrase is arguably the most complex aspect of the Maltese resource grammar. A lot of information needs to be propagated up to the VP level and potentially affixed to the main verb. The verb phrase is composed of the following parts:

```
oper
  VerbPhrase : Type = {
    v : {
      s : VForm => VerbStems ;
      hasPresPart : Bool ;
      hasPastPart : Bool ;
      } ;
    s2  : Agr => Str ;
    dir : Maybe Agr ;
    ind : Maybe Agr ;
    } ;
```

The `v` field essentially holds an entire verb record (without the `VerbInfo` which is no longer needed at this point). See section 2.2.2 above for a description of the verb record type. `s2` is the complement to the verb phrase with is typically a non-pronominal object. The `dir` and `ind` fields are used to indicate whether there are direct and/or indirect objects which need to be joined to the main verb. These fields can encode potentially non-existent information by using the `Maybe` type, which is modelled on the Haskell equivalent.

Adjectival phrase

The adjectival phrases are quite simple, containing a simple string which inflects for gender-/number and a boolean field `isPre` to indicate if the AP should go before or after a noun phrase.

```
lincat
  AP = {
    s : GenNum => Str ;
    isPre : Bool
  } ;
```

2.3.6 Clauses

The RGL covers three kinds of clauses: **declarative**, **question** and **relative** clauses, encoded with the categories `Cl`, `QCl` and `RCl` respectively. They are all implemented in similar ways:

```
oper
  Clause : Type = {
    s : Tense => Anteriority => Polarity => Order => Str
    } ;
  QClause : Type = {
    s : Tense => Anteriority => Polarity => QForm => Str
    } ;
  RClause : Type = {
    s : Tense => Anteriority => Polarity => Agr => Str
    } ;
param
  Order = ODir | OQuest ;
  QForm = QDir | QIndir ;
```

In each case, the clause needs to contain versions of the underlying statement for all tense and polarity combinations. Since question clauses and relative clauses can be made from declarative clauses, the `Clause` type needs to contain all possible word orders. This is the purpose of the `Order` parameter. Question clauses can also be linearised in different orders depending on whether they are direct or indirect.

Variable word order

Intonation in spoken Maltese can be very important in determining whether a statement is a declaration or a question. As a result of this, word order is relatively free and a phrase is a given order can be interpreted in different ways. Similarly, the same kind of phrase can be linearised in different word orders. Table 2.19 gives some examples of this variation.

Table 2.19: Examples of variable word order

Form	Order	Maltese	English gloss
Declarative	SVO	*Mary marret tgħum*	'Mary went swimming'
Declarative	VOS	*marret tgħum Mary*	'went swimming Mary'
Declarative	OVS	*tgħum marret Mary*	'swimming went Mary'
Question	SVO	*Mary marret tgħum?*	'Mary went swimming?'
Question	VOS	*marret tgħum Mary?*	'went swimming Mary?'
Question	OVS	*tgħum marret Mary?*	'swimming went Mary?'

Notwithstanding this, in the resource grammar we take a simpler approach and handle only one word order in each case: SVO for declaratives and VOS for questions. These are arguably

the most standard orders which are used most commonly. This of course has the effect that the grammar would be less effective for parsing open Maltese text.

2.4 Development and testing

2.4.1 Development tools

RGL source code browser

This web-based tool[5] allows the user to browse the source code of all modules in the RGL. Its main feature is the ability to search the scope of any module, thus allowing one to quickly reach the definition of a function which is inherited from some other module. This is achieved by using the `--tags` flag to the GF compiler, which produces so-called `.tags` files which contain all functions in the scope of each module. These files are processed and loaded by the RGL browser in an asynchronous fashion. The RGL browser was build to speed up resource grammar writing, by making it easy to look at implementations in other languages in order to inform one's own development.

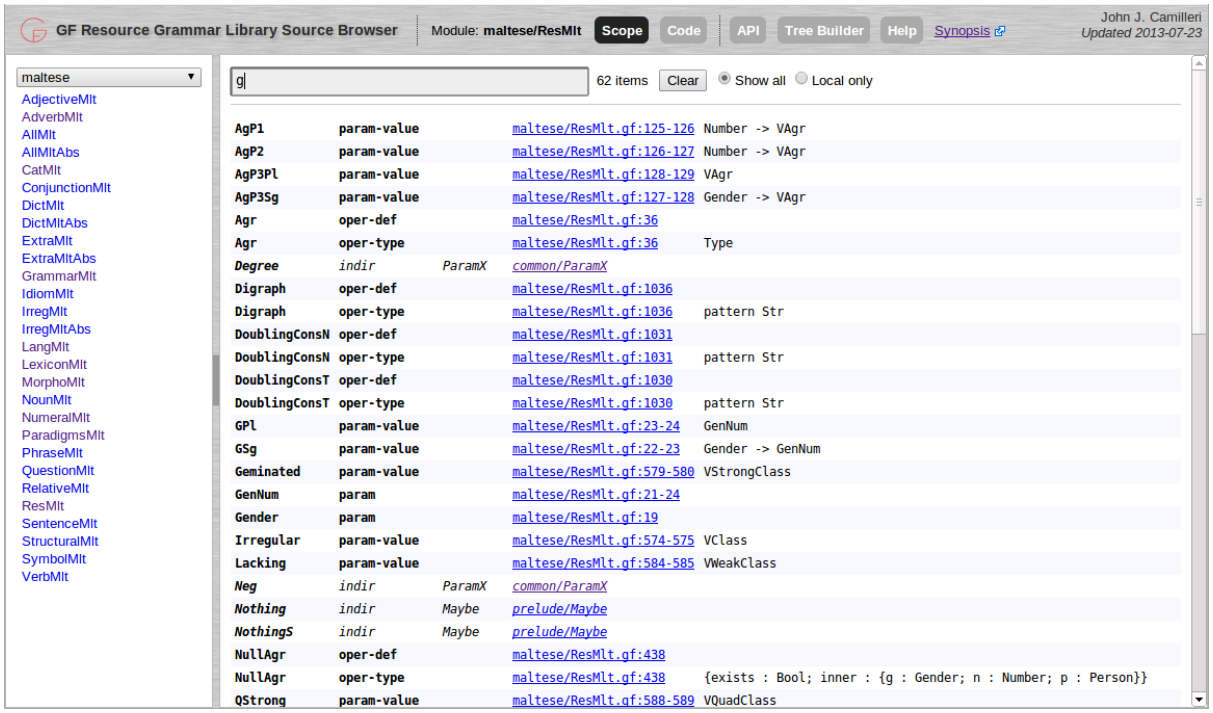


Figure 2.1: Screenshot of the RGL browser tool

Syntax tree editor

Another web tool developed during the work described in this thesis the syntax tree editor[6], which allows the user to construct abstract syntax trees graphically according to some grammar.

[5] http://www.grammaticalframework.org/lib/doc/browse/, accessed 2013-09-05
[6] http://cloud.grammaticalframework.org/syntax-editor/editor.html, accessed 2013-09-05

While designed primarily for application grammars, it can also be used for constructing API trees and testing them out in some reference language. Figure 2.2 shows a screenshot of this tool.

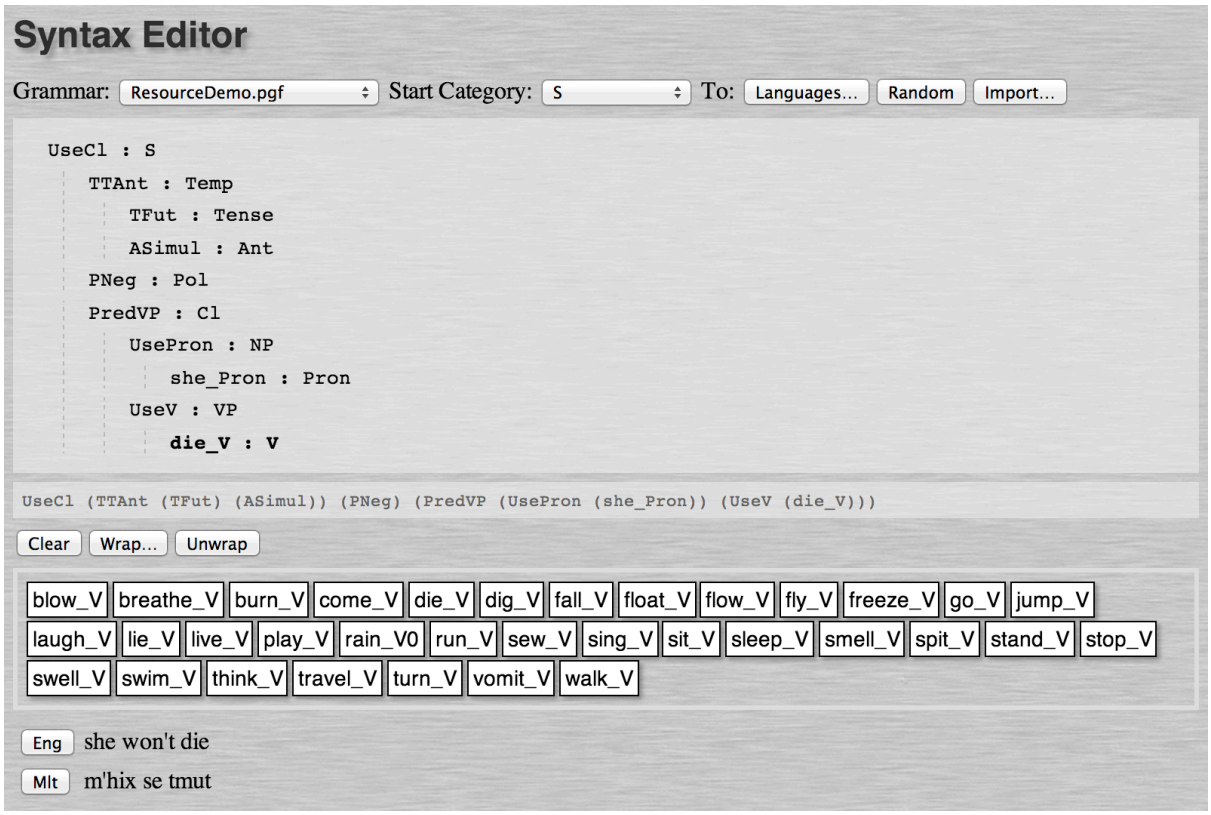


Figure 2.2: Screenshot of the syntax tree editor

2.4.2 Test methodology

The general methodology for testing GF resource grammars is as follows (Ranta, 2011):

1. For each feature / function being handled in the grammar, build up a list of abstract syntax trees (or API trees) which test this feature and write down their intended linearisations (gold standards).
2. This list can partially be built using the examples in the source code comments in each of the RGL's abstract syntax modules.
3. Write each function implementation to satisfy the gold standard, and check the actual linearisation produced by the grammar against the ideal one.
4. Each time a module is changed, re-run the regression test to ensure that no new bugs have been introduced.

In many cases it is helpful to choose one more **reference languages**, that is languages which have already been implemented in the RGL and which are familiar to the grammarian. Checking the linearisation of certain constructions in such a reference before implementing them in the current language is often useful. However, this comes with the proviso that resource grammars are not intended for general translation 'out of the box'. Rather, they should provide the means for an application grammarian to express what they want in a grammatically correct way. For this reason it can often be hard to judge whether the translation produce by a resource grammar is correct or not, as it would depend on the use of that grammar within the context of an application grammar.

2.4.3 Treebanks

A number of treebank files were written during development to test different specific aspects of the grammar. These treebanks are listed in table 2.20. In order to speed up the test-develop cycle, a simple script was written for linearising sets of test trees against the current grammar and indicating those those which are linearised incorrectly. The script is written in Haskell and uses the PGF Haskell library[7] which is provided as part of the GF distribution. The script, along with the treebank files themselves, can be found under the `test/regression/` directory of this work's source code repository (see appendix D).

The treebank files are written in a specific format which uses Org-mode syntax for plain text tables[8]. An example of this format is given below:

```
| AST                                                  | English           | Maltese         |
|------------------------------------------------------+-------------------+-----------------|
| DetCN (DetQuant IndefArt NumSg) (UseN airplane_N)    | an airplane       | ajruplan        |
| DetCN (DetQuant DefArt NumSg) (UseN airplane_N)      | the airplane      | l-ajruplan      |
```

Verb inflection treebank

In addition to the treebanks listed in table 2.20, it would also be relevant to mention a small treebank of Maltese verb inflections also compiled during this work. This treebank[9] consists of full inflection tables of 74 Maltese verbs of different paradigms, which have been tabulated in CSV format. This amounts to a total of 70,448 unique wordforms which have been manually checked. Unfortunately these could not be converted to the required format in the time available and are thus excluded from the results in the next section.

[7] http://hackage.haskell.org/packages/archive/gf/3.1.6.2/doc/html/PGF.html, accessed 2013-09-05
[8] http://orgmode.org/manual/Tables.html, accessed 2013-07-23
[9] Refer to appendix D for more details.

2.4.4 Coverage

Abstract function implementation

The following functions from the `NumeralTransfer` module were not implemented in the Maltese grammar:

```
active2passive dconcat digits2num digits2numeral dn dn10 dn100 dn1000 dn1000000a dn1000000b
    dn1000000c nd nd10 nd100 nd1000 nd1000000 num2digits
```

A bug in the GF compiler means that no warning is generated when these functions have no linearisation rules. However, these functions are in fact not implemented in **any** of the language implementations in the RGL and thus were not deemed essential.

Regression test results

Table 2.20 shows the results of running the treebank regression tests against the version of the Maltese resource grammar at the time of publication. As can be seen, a considerable number of linearisations (15%) do not match their gold standards. These failures can be generally attributed to one of the following:

- Irregular vowel changes in verbs (require more forms in the lexicon).
- Incorrect selection of stem form and/or suffix form when joining enclitic pronouns.
- Insufficient work on relative clauses.
- Errors in the use of the copula *kien* in some tense combinations.
- Bugs in the numerals module.

It is worth pointing out that the totals in the table give a measure to how well the grammar covers the treebank set itself; There is no measure however of how completely or evenly the treebanks cover the Maltese language in general. Note also that all the testing carried out is only evaluating *linearisation*; no testing is performed in the *parsing* direction.

Table 2.20: Treebank list and testing results

Name	Description	Trees	Passed	Rate (%)
`ap`	Adjectival phrases	2	1	50.0
`articles`	Articles	5	5	100.0
`exx-resource`	Wide-coverage treebank from `testsuite/libraries/` in the GF repository	186	111	59.7
`n-clitics`	Enclitic pronouns on nouns	49	35	71.4
`np`	Noun phrases	1	1	100.0

Continued on next page

`numerals-np`	Numerals as determiners	32	32	100.0
`numerals-simple`	Numerals in isolation	63	52	82.5
`phrases`	Declarative and question clauses	22	19	86.4
`prep`	Prepositions	24	24	100.0
`rel`	Relative clauses	4	0	0.0
`v-clitics-past`	Enclitic pronouns on verbs, past simultaneous (perfective)	392	336	85.7
`v-clitics-pres`	Enclitic pronouns on verbs, present simultaneous (imperfective)	392	368	93.9
`v-clitics-variants`		5	5	100.0
`vp`	Verb phrases	128	120	93.8
`vpslash`	Verb phrases with missing complements	2	2	100.0
			Total	85.0

Known shortcomings

Some other known shortcomings of the grammar which are not tested by the treebank tests are:

- Interaction with euphonic vowels and articles.
- *i*/*j* changes depending on preceding word (impossible in GF).
- Free word order

2.5 Problems encountered

2.5.1 Large inflection tables

The initial implementation of verb morphology was to handle enclitic pronouns in the inflection table of the verb. Combining inflections for subject, direct objects, indirect objects and polarity in a single table yielded an inflection table with roughly 1000 unique strings for each verb (see appendix C). This allowed very precise morphological control since compile-time strings can be pattern matched and modified.

However while this was technically possible, it led to problems with compile time for the grammar. The PGF format — to which GF grammars are compiled — stores all possible tokens individually, and having to compile record tables of this size started to become prohibitively slow.

As a result, the grammar had to be significantly refactored such that the verb inflection table was reduced to just stems, and all enclitic pronouns were suffixed via the bind operator at the syntax level. This is the standard approach to this problem in the RGL.

2.5.2 Binding of runtime tokens

Binding is a solution to the fact that (i) GF does not allow gluing suffixes to tokens at runtime, and (ii) storing hundreds or thousands of inflection forms in a table becomes very inefficient in terms of the time and space required to compile to PGF. This solution works by introducing a special bind token `&+` in between tokens which should be linearised as a single word during unlexing. This process of unlexing is deterministic and simple.

However, the lexing process — which requires the re-insertion of the bind token between stems and suffixes — can be problematic. Since this lexing must occur as a pre-processing step *before* input to the runtime, the lexer must essentially work without any access to the grammar rules defined in the GF modules. This means that without re-implementing potentially large parts of the morphological rules in an discrete lexer, ambiguities may arise. In Maltese for example, the suffix *-u* may be an enclitic pronoun indicating possession on a noun, e.g. *qalbu* ('his heart'), as well as a regular conjugational ending of a verb: e.g. *talbu* ('they prayed'). The former is composed of the stem *qalb* and the pronominal suffix *-u*, which should be lexed into tokens `["qalb","&+","u"]`. However a naïve external lexer may also try to lex *talbu* in the same way, which would be incorrect.

Thus is it clear that any lexing process must have at least some knowledge of the grammar itself, although the current GF implementation has no real support for this. As a result, without further developments in the runtime, this resource grammar is of very limited use for parsing. The same is true of other resource grammars which using this binding solution.

2.5.3 Pattern matching on runtime strings

The inability to pattern match or perform any query operations on strings at runtime can have a big effect on the design of a grammar. This becomes especially evident when stem changes occur during affixation, often referred to as **Sandhi**. In Maltese for example, the verb *fetħet* ('she opened') undergoes a vowel change from $e \rightarrow i$ when suffixed by an enclitic pronoun *-hom* to become *fetħithom* ('she opened them'). The joining of verb stems and suffixes occurs at the syntax level, for example in the `ComplSlash` function from the `VerbMlt` module. Yet by this point it is too late to analyse the verb stem and perform any required vowel changes. Instead, all possible verb stems must be computed within the morphological paradigm and stored in its inflection table. The selection of the correct stem must then be based on some other criteria, such as parameter values.

2.5.4 Non-existent forms

Partial functions

All functions in GF must be total; for example, if the function `PassV2` has type signature `V2 -> VP` then it must be defined for **any** term of type `V2`. It is possible to imagine situations where no passive verb phrase can be constructed out of a certain 2-place verb. Yet it is not possible to implement `PassV2` in such a way that it may only defined for *some* terms of type `V2`. This could

potentially be handled with dependent types on the abstract syntax level. However, since the RGL abstract syntax is language-independent this would be not be possible without losing some of this independence.

Incomplete inflection tables

It is often the case that certain slots in inflection tables simply do not exist in the language. A prime example in Maltese is the dual form, which only a small number of nouns actually have. When it exists, this form is used to replace both noun and determiner, e.g. *xagħarejn* ('two months'). For nouns with no dual form, one uses the numeral for 'two' together with the plural form of the noun, e.g. *żewġ fliexken* ('two bottles'). All objects of the same type must have the same linearisation type in GF. So if the linearisation type for a noun contains a slot for the dual form, what should this slot contain in the case of nouns with no dual?

The empty string `""` would seem to be the logical choice, however this can create parsing ambiguities if it is ever included in a linearisation. To prevent against this, syntax-level functions need to be able to check whether a noun indeed supports the dual form or not. But because of the limitations on runtime strings (described above), it is impossible to check, at runtime, whether a string is the empty string or not. Attempting to use some garbage string instead of the empty string does not help the situation at all, and adds the risk of appearing in input completion. The solution to this is to store this information in some non-string field, such as a boolean `hasDual` field, which can be consulted at runtime.

The `Maybe` type As this turned out to be a fairly common occurrence in the Maltese resource grammar, a special `Maybe` type was defined as a generalisation of this programmatic idiom. This type is defined as a type which takes another type as a parameter, as follows:

```
oper
  Maybe : (t : Type) -> Type = \t -> {
    inner : t ;
    exists : Bool
    } ;
```

Thus the type `Maybe Str` becomes a record containing a string and boolean field indicating whether the former exists or not. This type and its supporting operations can be found in the `Maybe.gf` module in the RGL's `prelude` folder. An unfortunate downside to using the `Maybe` type is that it causes a size explosion in the compiler when used in tables. This is explained below.

2.5.5 Parameters in tables

During the development of this resource grammar, it was discovered that using parameter fields within tables will cause an exponential explosion in the GF compiler. Consider the following example grammar:

```
concrete ParamConc of Param = open Prelude in {
```

```
  param P = P1 | P2 | P3 | P4 ;
  lincat
    S = Str ;
    A = P => P => {s:Str ; b:Bool} ;
  lin
    Start a = (a!P1!P1).s;
    It = \\ _,_ => {s="abc" ; b=True} ;
}
```

When compiling this valid GF module with `gf --verbose`, the following debug message is produced:

```
 + Start 65536 (1,1)
```

This indicates that the compilation of the `Start` function produces 65536 productions in the resulting PMCFG. This number fits with the number of possible functions of type `P` $\rightarrow$ `P` $\rightarrow$ `Bool`, which is $2^{4\times4}$. Removing the boolean field from the type gives instead $1^{4\times4}$ functions, and the size explosion disappears[10]. This behaviour means record types containing non-strings (such the `Maybe` type above) should not be used inside tables. Alternative factorisations need to be found in such cases, in order to avoid exponential explosions during compile time.

[10]Thanks to Thomas Hallgren for helping with finding a minimal example and explaining the arithmetic behind the explosion.

Chapter 3

Computational lexicon

This chapter begins by presenting a web application designed for collecting the heterogeneous lexical resources available for Maltese into a single database. After explaining the setup and implementation of this collection, we then go on to describe how it is combined with the resource grammar from the previous chapter to produce full-form computational lexicon.

3.1 Method

3.1.1 Sources

The approach adopted in this work for constructing a computational lexicon for Maltese is to first build a platform where all existing lexical resources can be gathered into a single collection. While there are some large, high quality print dictionaries available for Maltese (see section 1.2.1), the number and size of computational resources is only a fraction of this. Nevertheless, the hope is that an open platform for hosting and searching through resources from heterogeneous sources will be useful in its own right, and even attract the addition of new lexical resources that may become available in the future. The sources available at the time of writing were:

- An exhaustive list of all 4,142 root-and-pattern verbs (including hypothetical forms), from the verbal roots database (Camilleri & Spagnol, 2013).
- A corpus of 654 broken plurals for both nouns and adjectives (Mayer *et al.* , 2013).
- A list of over 2,500 verbal nouns listed in the Aquilina dictionary and other sources (Ellul, 2013).
- A Basic English-Maltese dictionary containing some 5,454 English entries (Falzon, 2012).

3.1.2 Heterogeneous data

Traditional relational databases work with a strict schema system, whereby the structure of all data is fixed at design time and all entries in the database necessarily conform to this schema. In this work however we are dealing with lexical resources from distinctly different sources,

where the data structure does not in any way match between one source and the next. Given a fixed list of sources, it is not difficult to design a relational schema which accommodates the data from each; in the worst case this typically results in a large table with lots of null fields. But this solution will be problematic if one wants to include data from new resources that may become available in the future.

In order to maximise adaptability to future resources, this collection of resources must be flexible enough to support heterogeneous data as it is. This will not only avoid the problem of having to design an inefficient schema which is the union of the structure of all the current resources, but more importantly will ensure that future data can be easily added to the collection without having to conform to any specific schema. More information about how this is achieved can be found below in section 3.2.1.

3.1.3 Full-forms

Traditional dictionaries are organised by head word or lemma, where different word forms for that lemma are specified merely as suffixes. Take the following entry from the Serracino-Inglott (2003, p. 218) dictionary as an example:

> **ħtieġa** *n.f.s., pl. -t, -ijiet ...*

The entry gives the singular form *ħtieġa* ('need') as a head word, together with the suffixes which give the plural forms. These affixes however cannot be blindly appended to the head word; the correct plural forms in this case are in fact *ħtiġiet* and *ħtiġijiet*. So even though the dictionary gives us some information about the other word forms for this lemma, it takes some further knowledge of the language in order to apply the rules correctly.

This is where the importance of having a full-form lexicon becomes apparent; even in cases where inflection is affix-based, other morpho-phonological rules can come into play. In cases where inflection is non-concatenative — which is often the case in Maltese — storing all inflected forms is essential if the lexicon is to be used for any kind of lookup or lemmatisation.

Storing all forms versus generating them

Two options exist when it comes to building a full-form lexicon:

1. All word forms are stored as individual entries in a database, all linked to the parent head word.

2. Only the headword itself is stored, and inflected word forms are produced in real-time by some automaton.

The former option is generally more demanding in terms of space requirements, yet the latter option depends on the morphological predictability of the language in question. In the case of Maltese, the prevalence of unpredictable broken plurals for nouns and adjectives is a clear indication that some way of storing full forms is needed, even if other plurals may be

more regular. For verb inflections, the morphological processes are in fact more regular and the argument in favour of only generating full forms becomes stronger.

One could imagine a system which combines both a morphological automaton together with a database of full forms as some kind override to the former. In many ways, GF lexicons defined using smart paradigms essentially work in this way. However GF itself is not really suitable as a final storage format for a lexicon because it essentially enforces a fixed schema on its entries, and extending it with new words from different sources may require considerable refactoring.

This work opts for the former of the options presented above, that is storing all forms in a single database, without the use of any real-time morphological generator. Apart from making the system design simpler, this means that the lexicon can more effectively searched (using regular expressions, for example) and that its contents can be more easily converted or exported to some other format.

Size calculations

Dalli (2002a) estimates that at least 30,000 lemmas must be identified in order to have significant coverage of all the Maltese language. As a comparison, Serracino-Inglott's Maltese dictionary (Serracino-Inglott, 2003) contains roughly 26,000 entries, while the Maltese-English volumes of Aquilina's (Aquilina, 1987, 1990) contain some 80,000. The total number of entries from the sources listed in the previous section amounts to almost 13,000. Note however that this does **not** take into account the duplicate entries appearing from different sources; the number of unique entries will therefore likely be lower.

Considering the worst-case inflectional forms of each of the major parts of speech, we have:

- Nouns have 5 plural forms, each of which can appear with or without enclitic pronouns, giving an upper bound of 40 word forms.
- Verbs have 952 forms (see appendix C) for the main moods/aspects, and 14 for present and past participles (which take no enclitic pronouns). This gives a total of 966 verb word forms.
- Adjectives have 3 inflection cases and 3 forms, making a total of 9 possible combinations.
- We will ignore inflections of other word classes such as prepositions and pronouns, as such structural words are generally of a fixed, small number.

In order to now estimate the total number of forms required over an entire lexicon, we must establish the distribution of the different parts of speech in a Maltese dictionary. An analysis of the digitised version of the Aquilina dictionary (MLRS, 2013) gives a worst-case total number of forms of 86 million. This calculation is broken down in table 3.1. While this figure is higher than Dalli's estimation of around 64 million unique wordforms (Dalli, 2002a, p. 69), it is certainly in the same order of magnitude. He goes on to estimate an average case of around 5.7 million unique wordforms (based on an "average word form" count of 72 (Mangion, 1999, p. 65)).

Table 3.1: Estimation of the number of word forms required in a full-form lexicon of Maltese. Counts for each part of speech are calculated from (MLRS, 2013).

Part of speech	Count		Word forms	Subtotal
Noun	198,237	(56.1%)	40	7,929,480
Verb	81,063	(22.9%)	966	78,306,858
Adjective	49,853	(14.1%)	9	448,677
Other	24,361	(6.9%)	1	24,361
Total	353,514	(100%)		86,709,376

3.2 Application

This section details the design and implementation of a database for collecting together the available lexical resources for Maltese, along with the web application for interfacing with this database, named *Ġabra* (literally, 'collection'). Details on accessing this application can be found in appendix D.

3.2.1 Database

Design

As discussed previously, a central principle in the database design for this collection is that it maximises flexibility and can accommodate heterogeneous data. To this end, a very general top-level structure is needed which should minimally accommodate the following:

1. A separation between lemma and wordform levels with a one-to-many relation from the former to the latter.
2. Arbitrary feature names and values for each wordform and lexeme.
3. Having a distinct entity for consonantal roots, which are shared by multiple entries of different lexical categories.
4. Links between lexemes which can be given arbitrary relation names and properties.
5. Attribution of any kind of entry to a source with associated licensing information.

The top-level structure adopted for this database is show in figure 3.1. **Lexemes** are the entries as we think of them in a dictionary. Each is represented minimally by a headword and POS tag, but can often include a gloss in English and other features common to the entire entry. **Wordforms** represent the inflected forms within a lexeme, and include the lemma as wordform. They only need to consist of a single string, but ideally will list the features that they inflect for. The **roots** are modelled as separate entities as they are shared by multiple lexemes. It could

be argued that this separation is unnecessary, and all the root information can be stored in the lexemes (de-normalised).

In addition to the above, all entries optionally have a `source` field which indicates the citation reference of the source of the entry. This exists so that the information stored in the database does not become disconnected from its original authors (the sources entity is not shown in the schema diagram).

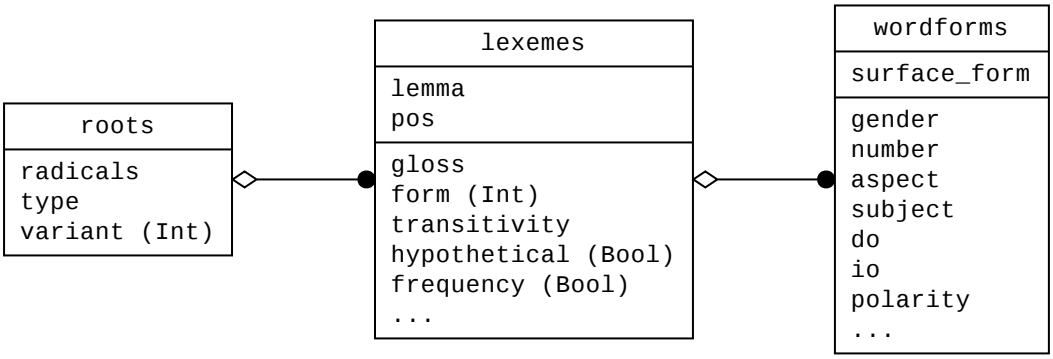


Figure 3.1: Basic data schema used in the lexicon. Each box shows (in order) the entity name, the fields that are always present, and optional fields that only occur in some documents. All fields are of type string unless marked. Primary and foreign key fields are omitted.

Realisation

The database engine chosen for implementing this lexicon is MongoDB[1], a document-oriented system which stores data as JSON-style documents[2] with dynamic schemas. The great advantage of this system is that documents in the same collection can have different structures, which is exactly what is needed in this project. This makes it very easy to add new features to any entry without changing the table schema and ending up with lots of null fields.

A side effect of this however is that the database engine does not itself handle joins, as with a traditional RDBMS. Thus when modelling data using MongoDB, data de-normalisation is often recommended[3]. This means that rather than splitting data between multiple tables and depending on foreign keys and joins, the same data may be stored in multiple locations. This introduces some overhead in maintenance and introduces a risk of data inconsistency, but makes data retrieval and searching easier and faster.

[1] http://www.mongodb.org/, accessed 2013-09-05

[2] Note about MongoDB nomenclature: a "collection" can be thought of as a table, a "document" is an entry in a collection, and a "field" is analogous to a table column. The concept of a schema is not enforced at all, and the term is used loosely.

[3] See http://docs.mongodb.org/manual/core/data-modeling/, accessed 2013-09-03

Importing data

Importing of data into the database was carried out using a number of scripts written in Haskell and using the `mongoDB` Haskell library[4]. The general steps in each case are described below.

Verbal roots project database This data was available as a MySQL database, with one table for roots and another for verb forms. The import script made use of the `mysql-simple` Haskell library[5] to query the database directly and insert the data into MongoDB.

Broken plurals This resource is made available in tab-separated format, which was straightforward to process. Each line of the original file contained a new form. In cases of adjectives, two singular entries would have the same plural e.g. *abjad* (m.sg.) and *bajda* (f.sg.) are separate entries which both have the plural *bojod* ('white'). The import script therefore detects these cases and labels them as adjectives. However this can fail with animate nouns which also have two genders, e.g. *tabib* ('male doctor'), *tabiba* ('female doctor') and *tobba* ('doctors'). In these cases the POS needs to be corrected manually since the source gives no other information to help disambiguation.

Verbal nouns In this case, the word list was only available as a Microsoft Word document containing a table spanning many pages. As this format is less amenable to direct extraction, the data was first copied-and-pasted into a text editor. This step eliminated the table structure by placing every cell on a new line, which could then be more easily processed using standard techniques. A disadvantage of this step is that all formatting information was lost in the conversion. In particular, entries formatted in italics indicating hypothetical words could no longer be distinguished.

Basic English-Maltese dictionary This dictionary has been made available in an XML format and therefore highly amenable to automatic processing. The import script for this resource uses the `xml` Haskell library[6] for parsing XML. Unlike each of the importation steps described above, which work in isolation and provide non-overlapping sets of data, the importation of an entire dictionary must be able to handle duplicates. That is, if the lemma being imported already exists in the collection then their data should be merged, rather than a duplicate entry being created. The script runs in a batch mode but keeps logs about what entries were merged, which must be checked manually.

[4] http://hackage.haskell.org/package/mongoDB, accessed 2013-09-01
[5] http://hackage.haskell.org/package/mysql-simple, accessed 2013-09-01
[6] http://hackage.haskell.org/package/xml, accessed 2013-09-03

3.2.2 Web application

CakePHP framework

CakePHP[7] is an open-source web development framework written in PHP. It uses a MVC (model-view-controller) architecture to cleanly separate data, business logic and user interface. The framework is typically used with relational databases, however a data source implementation exists which allows it to be used with MongoDB[8].

User interface

Browsing Figures 3.2 and 3.3 show screenshots of the online lexicon application. The former shows how one can browse and search by root and derivational form, which aims to replicate the same interface as in the online database of root and pattern verbs (Camilleri & Spagnol, 2013). The latter is an example of searching through the lexeme entries, which is the primary way in which a lexicon is usually used. Figure 3.4 shows the detail page for a single entry, where all the wordforms for that lexeme are displayed and can be filtered.

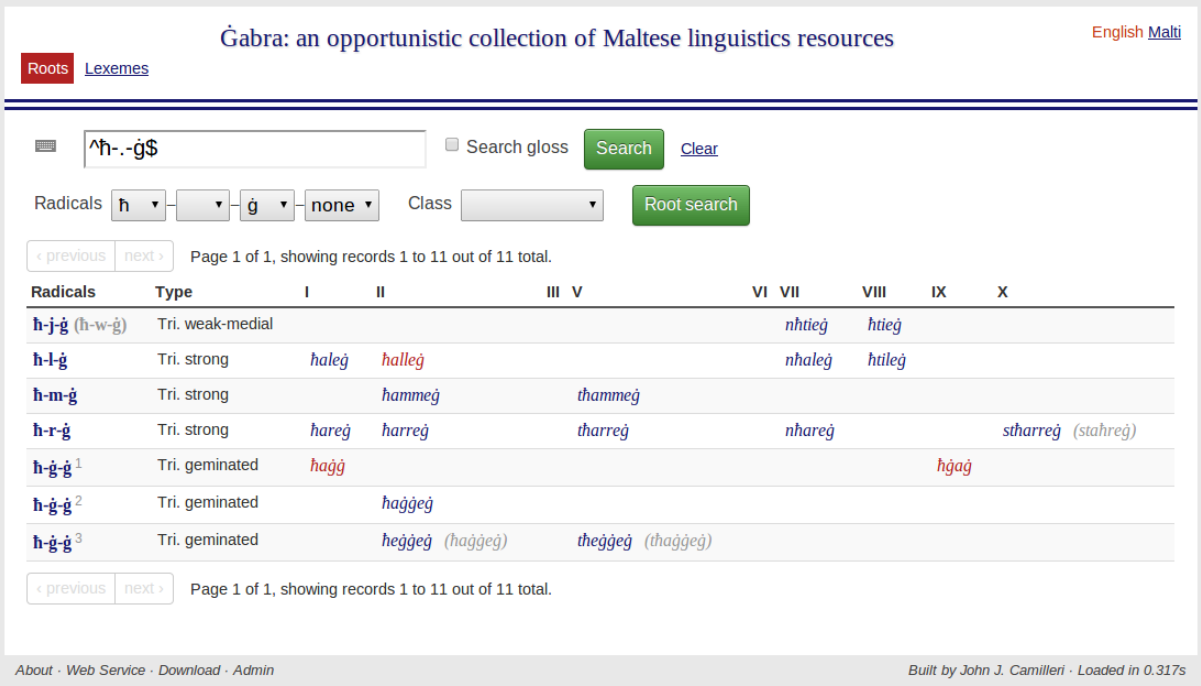


Figure 3.2: Searching for a root by regular expression in the computation lexicon

Searching All search terms in the application are treated as regular expressions. A regex helper interface is provided on roots page to help users unfamiliar with regex syntax (shown being used in figure 3.2). In order to prevent against slow database queries, the beginning of string anchor ^ is inserted automatically before all queries. This a merely a user-interface

[7] http://cakephp.org, accessed 2013-09-03
[8] https://github.com/ichikaway/cakephp-mongodb/, accessed 2013-09-03

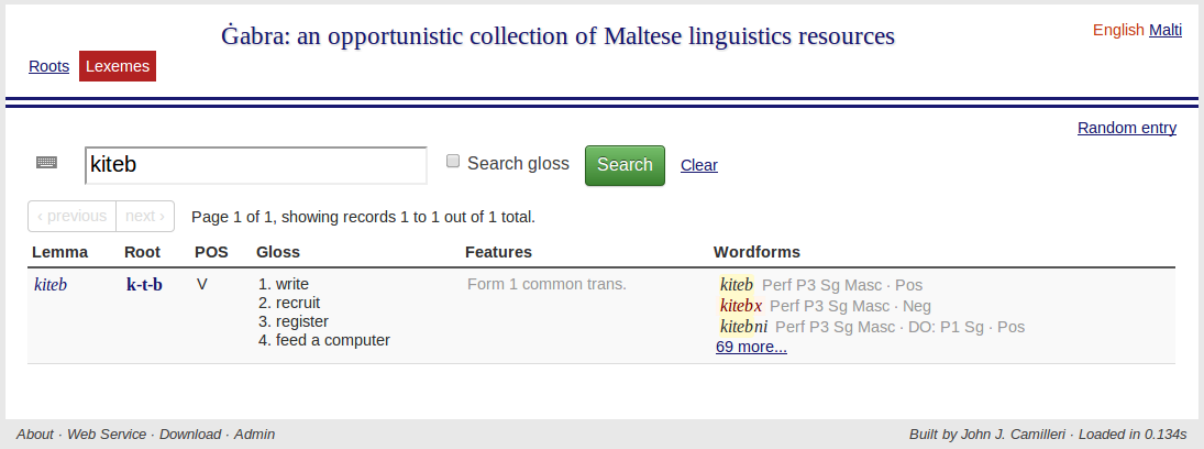


Figure 3.3: Searching for a wordform by substring in the computation lexicon

restriction; the database engine itself allows any regex query to be run. Any search term given is looked for in all the wordforms in the system, or optionally in the English glosses if one one wishes to use the lexicon for translation.

Feedback As mistakes may be made during the importation or automatic inflection processes, it is useful to include a system for users to report any errata or omissions they may come across. All wordforms displayed in the web application are accompanied with a small icon which can be clicked to instantly report an entry as incorrect. These reports can then be checked manually and corrected as necessary. Wordforms already marked as incorrect and pending review are marked in red, as shown in figure 3.4.

Apart from reporting errors, the application also includes a system for users to suggest new additions to the collection. This option shows up when somebody searches for a term which produces no results. Submissions are also placed in a moderation queue which must be checked manually by the maintainers.

3.2.3 Web service

Apart from a user interface, this web application also includes a web service to facilitate accessing the resource from from other scripts/applications. The response format can be either JSON or XML. The API for the web service is still considered as beta and will likely change if/when the web service begins to be used by other developers.

3.2.4 Issues

Loading data

Even with no joins, loading large amounts of data can sometimes be slow, especially over a network. To improve the user experience, the loading of data is often performed asynchronously within the application. This does not decrease the total amount of time required (it may

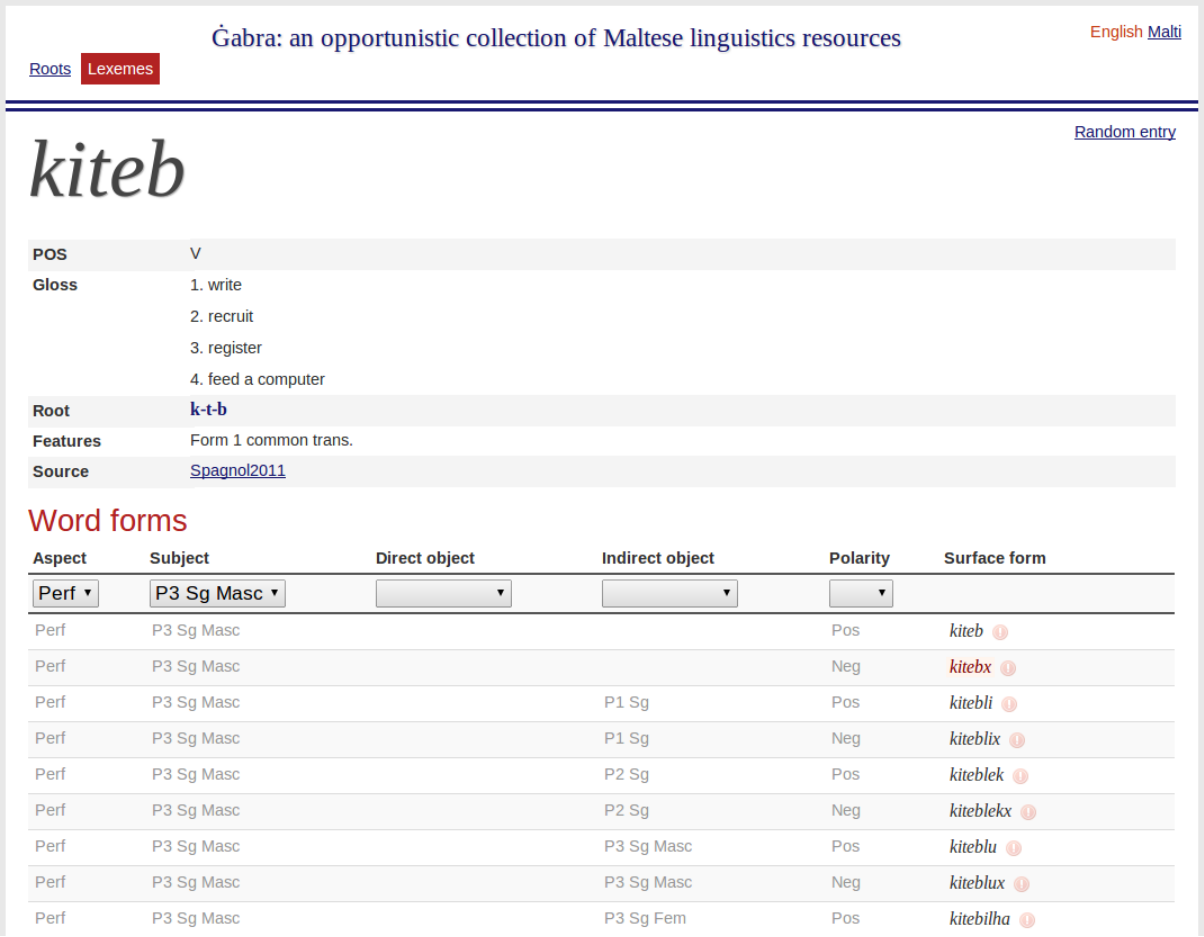


Figure 3.4: Viewing the details of a particular entry, where one can filter through the wordforms

even increase it slightly), but giving the user at least some information quickly feels a lot better than making them wait for a long time and then displaying it all at once.

Search performance

In our application most searches are carried out on the `wordforms.surface_form` field. Despite containing over 4 million forms, using a database index on the field makes searching time very good. In most cases the response time is under 100ms, even when searching using regular expressions. Yet some queries do take much longer to complete (in the order of seconds), generally when the beginning-of-string anchor ^ is omitted. Table 3.2 gives some example queries and their response times.

Table 3.2: Example database queries and response times

Type	Query	Response time (ms)
Basic text search	`db.wordforms.find({"surface_form":"skrejjen"})`	< 100

Continued on next page

Anchored regex	`db.wordforms.find({"surface_form":/^skrej/})`	< 100
General regex	`db.wordforms.find({"surface_form":/skrej/})`	~ 4000

One could use an alternative engine for searching, e.g. Apache Solr[9]. However the response times in the application are generally acceptable and do not warrant a more heavy-duty solution.

Native sorting

MongoDB has a `sort` command which uses an inbuilt sorting algorithm for ordering documents by some field. Specifically, when sorting Unicode strings the database engine will sort in order of binary representation. Apart from not being able to perform case-insensitive searches, this also means that one cannot search according lexicographically according to the Maltese alphabet; the letters *ċ*, *ġ*, *ħ* and *ż* are incorrectly sorted after the letter *z*, and *għ* and *ie* are not correctly treated a digraphs.

This means, for example, that *ġara* ('neighbour') would be sorted after *zuntier* ('churchyard'), though it should come before it. Since the sorting scheme does not support digraphs, a word containing the *għ* such as *għaraf* ('he recognised') is erroneously sorted after *gara* ('he threw'). Sorting by custom collations is not currently supported by the MongoDB engine, however it has been marked a planned feature[10]. Until this is implemented, sorting can instead be performed at the application level. The disadvantage with this is that it excludes the possibility of using the database engine to efficiently provide pagination by combining the `sort()` and `limit()` commands.

3.3 Monolingual GF dictionary

With all the lexical entries gathered together in a computational lexicon, a monolingual GF grammar module can be easily constructed. In following the RGL convention, this consists of matching abstract and concrete modules named `DictMltAbs.gf` and `DictMlt.gf` respectively. These are included with the rest of the Maltese resource grammar described in chapter 2.

3.3.1 Method

The method for generating a monolingual GF dictionary from a word list is quite straightforward:

1. For each lemma in the lexicon, a valid GF function identifier is generated which is guaranteed to be unique. This is often an ASCII-ised version of the lemma combined with some unique identifier and suffixed with the POS. In the case of verbs, for example, we

[9] http://lucene.apache.org/solr/, accessed 2013-08-27
[10] https://jira.mongodb.org/browse/SERVER-1920, accessed 2013-08-27

include the derived form number and root radicals where relevant. Thus *rikeb* ('he rode') becomes `rikeb_RKB_1_9451_V`.

2. A function is defined in the abstract syntax module for this identifier, with a category to match its POS.

3. A linearisation for this function is defined in the concrete syntax module, which uses a smart paradigm for the particular POS and supplying (minimally) a lemma form. In the case of Maltese verbs, the root is also supplied since this information is available in the lexicon.

At the end of this process one has a pair of modules which together constitute a monolingual dictionary in GF. As with the rest of a resource grammar, such modules are typically not used directly but imported in other application grammars as required.

Quality

It is common to require some post-editing of the generated dictionary grammars since invariably in some cases the wrong paradigm will be chosen. As dictionaries are often quite large however, this might turn into a somewhat time-consuming task.

While GF smart paradigms are generally designed to work with minimal information, all relevant data that exists in the source word list can and should be be used in order to improve the accuracy of the smart paradigm. Specifically, valency information would help in order to choose between the correct forms of each lexical category, such as `V, V2` etc. Unfortunately only minimal amounts of valency information is available in the given sources and thus can only be made use of to a small extent.

3.3.2 Dictionary modules

A monolingual dictionary in GF has function identifiers which are not shared with other languages. For this reason, the dictionary must consist of both an abstract and concrete module. The following is an example from each:

```
abstract DictMltAbs = Cat ** {
  ...
  fun rikeb_RKB_1_9451_V : V ;
  ...
}

concrete DictMlt of DictMltAbs = CatMlt ** open ParadigmsMlt in {
  ...
  lin rikeb_RKB_1_9451_V = mkV "rikeb" (mkRoot "r-k-b") ;
  ...
}
```

3.4 Generating full forms

After creating a monolingual dictionary module as described in the previous section, this module can be used with the GF runtime in order to produce full forms for each of our lexemes. This task is also carried out using a Haskell script, which accesses a compiled version of the dictionary modules using the PGF library.

3.4.1 Nouns

For nouns which do not already have an explicit plural form included in the lexicon, just the use of the smart paradigm alone will provide a plural form. This can be found by using the `linearise -table` command.

In cases where nouns take a postclitic pronoun to indicate possession, one cannot simply look at its inflection table since clitics are handled at the syntax level. Instead, a template tree is constructed and linearised with the `bind` lexer in order to produce the correctly inflected form. For nouns, the following template tree was used:

```
DetCN (DetQuant (PossPron <DO>) NumSg) <N>
```

where `<DO>` is the pronoun indicating the direct object (possessor) and `<N>` is the noun under analysis.

3.4.2 Adjectives

For adjectives, simply using the `linearise` command on the lexical units is enough to get the inflections for gender and number, and the forms for comparative and superlative.

3.4.3 Verbs

For verbs, the situation is a lot more complex, Since a number of morphological features are produced by syntactic constructors in the grammar, we again need to use template trees for producing inflections. At the top level, sentences in the different tenses and polarities are generated using these templates:

```
Perfective:   UseCl (TTAnt TPast ASimul) <Pol> (PredVP (UsePron <Subj>) <Cl>)
Imperfective: UseCl (TTAnt TPres ASimul) <Pol> (PredVP (UsePron <Subj>) <Cl>)
Imperative:   UttImpSg <Pol> (ImpVP <Cl>)
              UttImpPl <Pol> (ImpVP <Cl>)
```

where `<Cl>` is a clause, which may be defined in many ways depending on what combination of object clitics is being produced:

```
Direct:  UseV <V>
DO:      ComplSlash (SlashVa <V>) (UsePron <DO>)
IO:      AdvVP (ComplSlash (SlashVa <V>) <NP>) (PrepNP for_Prep (UsePron <IO>))
DO + IO: AdvVP (ComplSlash (SlashVa <V>) (UsePron <DO>)) (PrepNP for_Prep (UsePron <IO>))
```

The meta-variables in the above trees are summarised below:

Placeholder	Meaning	Example
`<CN>`	Common noun	`UseN foot_N`, `UseN2 father_N2`
`<Pol>`	Polarity	`PPos`, `PNeg`
`<Subj>`	Subject	`i_Pron`, `youPl_Pron`
`<Cl>`	Clause	(see above)
`<V>`	Verb	`go_V`
`<DO>`	Direct object pronoun	`i_Pron`, `youPl_Pron`
`<IO>`	Indirect object pronoun	`i_Pron`, `youPl_Pron`
`<NP>`	Noun phrase	`DetCN (DetQuant DefArt NumSg) (UseN door_N)`

Speed

While the scripts are relatively straightforward, the one for verbs in particular can take a considerable length of time to finish running. This happens simply because there so many trees need to be linearised by the grammar. Running the full-form script on the four thousand verbs available in the collection yielded the following:

```
time runghc wordforms.hs

real  293m38.611s
user  282m50.424s
sys   2m51.348s
```

Chapter 4

Conclusions

This final chapter ends the thesis by highlighting the most major contributions of this work, together with its limitations. After a brief summary of related work in this field, we conclude with some directions for future work.

4.1 Contributions

All in all, the goals set out in the beginning of this thesis have all been met successfully. The main contributions of this work can be summarised as follows.

1. The first computational grammar of Maltese covering both syntax and morphology, and which is freely available as open-source software.
2. A new language implementation for the GF Resource Grammar Library. Maltese is the 28^{th} language to be added to the library, and the first complete implementation of a Semitic language. It is also the 11^{th} resource grammar to have a monolingual dictionary module.
3. An online collection of digital lexical resources for Maltese, which is searchable in a unified way and whose contents is freely available. It is hoped that this collection platform can become the first complete full-form computational lexicon for Maltese.

4.2 Limitations

4.2.1 Grammar limitations

Coverage

Despite all the work put into the development of the resource grammar, there are a number of known limitations in terms of coverage of the Maltese language.

- Interactions at word boundaries do not work perfectly. Some phenomena, such as the insertion of euphonic vowels with the definite article are incomplete, while others such

as the *i*/*j* switching depending on preceding words are not handled at all due to limitations in GF.

- Sandhi rules, which determine stem changes when enclitic pronouns are suffixed to words are encoded but there are known exceptions which have not been handled.
- Maltese' free word order is not captured at all; clauses of a particular type have a fixed word order.
- Relative clauses have not been extensively tested.
- The current version of the grammar does not pass all the treebank tests created during the development (see table 2.20).

Evaluation

In addition to known issues in coverage above, it also relevant to mention limitations in the evaluation process itself.

- The treebanks are developed manually to test known critical areas of the grammar, but there is no measure of how completely they cover the actual Maltese language. Coverage of the RGL abstract syntax and coverage of the target language are two different things.
- The calculation of treebank scores is very simply defined as the percentage of successful linearisations. This means that these scores are essentially skewed by the number of test trees written to cover each feature of the language. It is likely that in the current treebank set, verb inflections are covered more extensively than any other aspect of Maltese.
- A typical way to qualitatively test a resource grammar is by using it in an application grammar for some particular use case. This was not carried out at all as part of this work for lack of time.

4.2.2 Lexicon limitations

Database

- As mentioned in section 3.2, the chosen database engine does not allow sorting by a custom collation, which in the case of Maltese means that results shown to the user might not be sorted as they would expect.
- Despite using database indexes on searchable fields, open regex searches can become a little slow to execute, depending on the search term used. To alleviate this, all regex searches received from the web application are prefixed with the beginning-of-string anchor ^. This reduces slightly the search capabilities offered to the user.
- No versioning or auditing of any kind is kept by the database engine, which means that any changes made are irreversible. This introduces a risk of data corruption which is only mitigated against by having regular database backups as data dumps.

Web application

- The user interface for the web application was designed in an ad hoc manner and no real study or research was performed into user interfaces for online lexica. No tracking of website usage is recorded by the application, so there is thus no way of analysing usage in order to improve the interface.
- The user facilities for reporting mistakes and adding new contributions to the lexicon are completely anonymous and there is no protection against spam. Requiring users to register and login before providing feedback would increase the number of steps required and likely reduce the amount feedback received.
- A single bug in the morphological grammar can produce hundreds or thousands of incorrect word forms. Firstly, this may be distracting to users who may try to flag each of them individually (with no added benefit). Secondly, no method has been put in place for regenerating incorrect word forms after making a change in the grammar.

4.3 Related work

4.3.1 Computational linguistics resources for Maltese

Much of the background for Maltese computational linguistics — in particular the Maltilex project — is discussed in section 1.2.3. This section mentions some more disparate works which are or less direct relevance or use to this work. A useful background on some of the past work in Maltese computation linguistics can be found in (Dalli, 2002a, section 2.5).

The earliest mentioned morphological analyser for Maltese seems to been written by Galea (1996) in an undergraduate project entitled "Morphological Analysis of Maltese Verbs". Unfortunately neither the source code nor the publication itself were available at the time of writing.

Another verb morphology generator for Maltese was written by Ramon Casha from the Malta Linux User Group. The work is unpublished and was obtained via personal communication with the author. While it is known to have been used to generate a `aspell` dictionary for Maltese[1], no metrics for its coverage could be found. A continuation of this work has recently been undertaken by Mike Vella[2].

A proprietary online English-Maltese dictionary is maintained by Ian Vella[3], containing some 12,219 English entries. The dictionary contains no other lexical information.

Methods for building a computational lexicon for Maltese based on corpus extraction are covered at length in (Dalli, 2002b). The primary difficulty with this approach is that stemming or lemmatisation in a language with a rich, non-concatenative morphology is especially tough.

Dalli *et al.* (2004) describes the use of web services for linguistic resources, focusing mainly on system architectures and communication protocols. Web service technologies have matured

[1] ftp://ftp.gnu.org/gnu/aspell/dict/mt, accessed 2013-09-04
[2] https://github.com/vellamike/maltese-spell-checker, accessed 2013-09-04
[3] http://www.englishmaltesedictionary.com/mt.htm, accessed 2013-07-18

considerably since this paper was written, and recommendations therein are no longer terribly relevant.

A standard tag-set for the tagging of textual documents in Maltese has been developed by Gatt *et al.* (2003). This tag list, which is used in the annotation of the second version of the MLRS corpus can also be found on the MLRS website[4].

4.3.2 GF resource grammars

An undergraduate project at the University of Malta by Zammit (2012) also looked at the implementation of a GF resource grammar for Maltese. For lack of time, that project focused mainly on clause-level syntax. Some contributions from that piece of work have helped the design process of the current version of the grammar. However the source code from this work was not made available and thus no substantial effort at combining code bases was possible. The source code presented in this work is all original, and has only been contributed to conceptually by Zammit.

GF resource grammars exist for numerous languages of all language families and can be found in various stages of completion. A complete list of publications associated with the other languages in the GF resource grammar library can be found at the Grammatical Framework website[5].

4.3.3 Grammar formalisms

Numerous grammar formalisms have developed in the past few decades which are based on the natural language theory that the grammatical constituents are distinguished a lexical arguments or functions from one set of arguments to another. This class of "lexicalised" theories of grammar are called categorial grammars.

Tree-Adjoining Grammar (TAG) (Joshi & Schabes, 1997) is a formalism similar to context-free grammars, which describes the rewriting of trees rather than symbols. The XTAG project[6] uses lexicalised TAGs to build a wide-coverage grammar for English. It also serves as a grammar development system including a parser and morphological analyser.

Combinatory Categorial Grammar (CCG) (Steedman & Baldridge, 2005) is an efficiently parsable and linguistically expressive grammar formalism which relies on combinatory logic. It is another type of phrase-structure grammar, which generates constituency-based structures.

Lexical Functional Grammar (LFG) (Kaplan & Bresnan, 1995) is a framework for phrase structure grammars which focuses mainly on syntax and its relation to morphology and syntax. The Pargram project[7] is an international collaboration pf LFG-based grammar and semantics development. It aims to produce wide coverage grammars for a number of languages written in LFG and with a commonly-agreed-upon set of grammatical features.

[4] http://mlrs.research.um.edu.mt/index.php?page=34, accessed 2013-06-09
[5] http://www.grammaticalframework.org/lib/doc/rgl-publications.html, accessed 2013-09-04
[6] http://www.cis.upenn.edu/xtag/, accessed 2013-09-04
[7] http://pargram.b.uib.no/, accessed 2013-09-04

Head- driven Phrase Structure Grammar (HPSG) (Pollard & Sag, 1994) is a highly lexicalised generative grammar theory, where the lexicon has a richly structured hierarchy of types. In most applications, HPSG grammars are used as stand-alone wide-coverage parsers. **Sign-Based Construction Grammar** (SBCG) (Sag *et al.* , 2012) is a more recent formalism based on HPSG. The **LinGO Grammar Matrix** (Bender *et al.* , 2002) is an HPSG-based framework for the development of broad-coverage, precision, implemented grammars for diverse languages. LingGO Matrix grammars are available under open source licenses.

CCGs, TAGs and Head Grammars have been shown to all be weakly equivalent to each other, in the sense that they all define the same string languages (Vijay-Shanker & Weir, 1994).

Regulus (Rayner *et al.* , 2006) is a multilingual platform for compiling unification grammars for speech-based applications. It provides open-source resource grammars for a handful of languages.

Functional morphology (FM) (Forsberg & Ranta, 2004) presents a method for implementing natural language morphology in Haskell, which has been applied to small number of languages. It can be thought of as a fragment of GF embedded in Haskell, yet some phenomena such as stem-internal vowel changes are easier to write in FM due to Haskell's more powerful string and list processing functions.

4.3.4 Rule-based translation

Apertium[8] is an open-source system for rule-based machine translation (RBMT). Unlike GF, its rules are transfer-based and there is no language-independent representation of meaning. Partial implementations exist for translation from Maltese to Arabic[9] and Hebrew[10], but these could not be used in any tangible way in the current work.

GramTrans[11] is a cross-platform machine translation platform which is also transfer based. It offers free web-based translation for the Scandinavian languages, but the source code is proprietary.

4.4 Future work

4.4.1 Grammar extensions

- The primary task that this work leads to is of course increasing the coverage of the resource grammar to include the failed cases from the treebank tests, and the other limitations in the grammar noted in section 4.2.1. The treebanks accompanying the grammar should also be extended accordingly.

- The `Extra` abstract module distributed with GF is not a part of the RGL API but contains a number of constructions which languages can optionally implement. Many of these extra

[8] http://www.apertium.org/, accessed 2013-07-24
[9] http://sourceforge.net/p/apertium/svn/46324/tree/staging/apertium-mt-ar/, accessed 2013-07-24
[10] http://sourceforge.net/p/apertium/svn/46324/tree/staging/apertium-mt-he/, accessed 2013-07-24
[11] http://gramtrans.com/, accessed 2013-09-04

constructions apply to Maltese too, such as verb phrase conjunction via the `VPI` category.

- If the resource grammar is to be used for parsing of open-domain text, a specific parse grammar should be written which augments the current resource grammar with variants for spelling and word orders.
- Perhaps the best way of testing the robustness of a resource grammar is to use it as a library in some domain-specific application grammar. Numerous application grammars in GF have been developed over the years in various sizes, and adding Maltese versions of these grammars can serve as a quick and fun way of testing the behaviour of the resource grammar.

4.4.2 Lexing solution

The bind solution for affixation in GF is described in section 2.5.2. This solves a problem when linearising text, however introduces a problem when attempting to parse. These bind tokens need to be re-introduced during lexing, but this is not necessarily a trivial task. To date, no satisfactory and general solution exists for this problem within GF, which means that grammars using binding for affixation can perform very poorly when it comes to parsing.

Having a solution to this lexing problem would benefit many of the languages in the RGL. Ideally this process should be handled internally by the GF runtime itself, using a grammar's own rules to extract a lexer. Solving this within the GF runtime would mean that it would instantly benefit all languages using binding.

4.4.3 A linked Maltese WordNet

Given a computational lexicon with English glosses, one could use this for reverse-lookup and attempt to create a linked version of the Princeton WordNet. This would involve looking up each entry from the English WordNet in the Maltese lexicon and using each result as a translation. Some work on using linked WordNets to bootstrap large-scale GF dictionaries has already been carried out with Bulgarian, German, Finnish and Hindi, with varying levels of success (Virk, 2013, ch. 5).

Appendix A

Resource grammar structure

A.1 Categories

Figure A.1 shows the categories defined in the Resource Grammar Library and how they fit in the generalised linguistic hierarchy defined by the library. Table A.1 describes these categories, giving examples in English and Maltese for each.

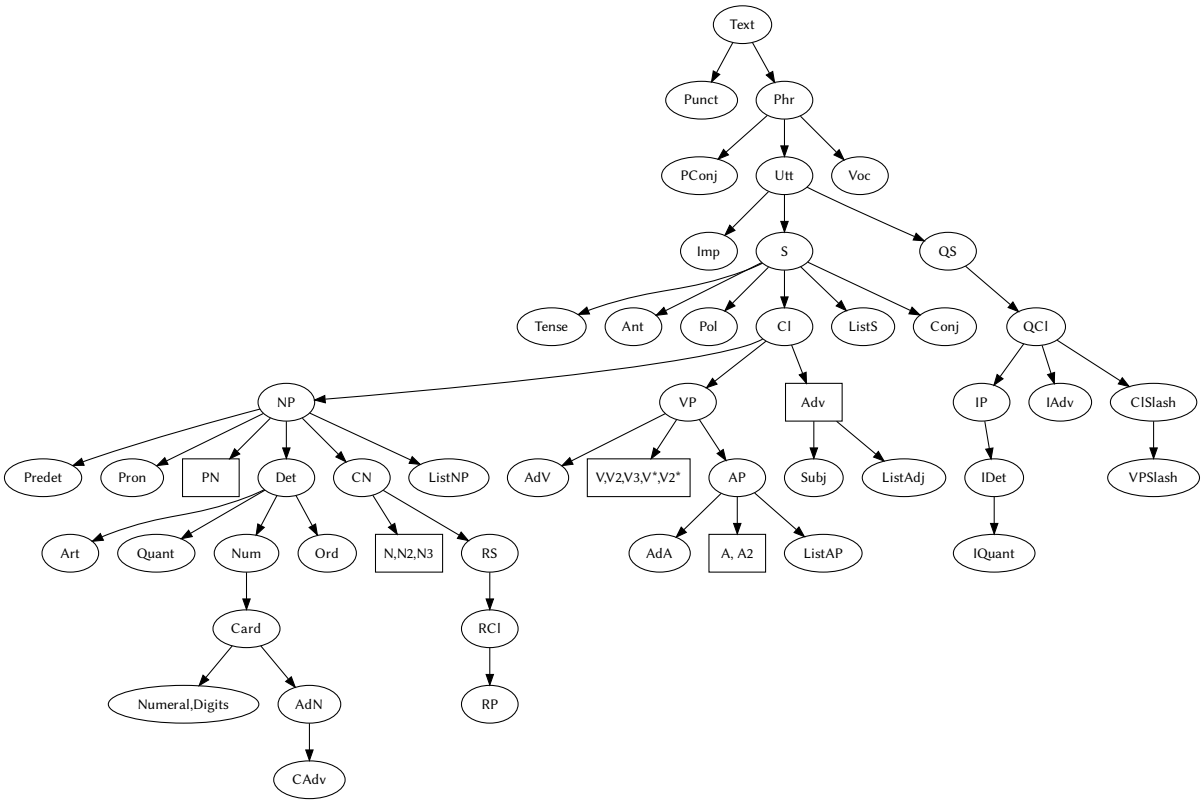


Figure A.1: Principal dependencies of phrasal and lexical categories in the RGL. Lexical categories appear in boxes rather than ellipses. Taken from http://www.grammaticalframework.org/lib/doc/synopsis.html

Table A.1: Description of the RGL categories, with examples. Taken from http://www.grammaticalframework.org/lib/doc/synopsis.html

Category	Explanation	English	Maltese
`A`	one-place adjective	*warm*	*sħun*
`A2`	two-place adjective	*married*	*miżżewġ*
`AP`	adjectival phrase	*very warm*	*sħun ħafna*
`AdA`	adjective-modifying adverb	*very*	*ħafna*
`AdN`	numeral-modifying adverb	*at least*	*mill-inqas*
`AdV`	adverb directly attached to verb	*always*	*dejjem*
`Adv`	verb-phrase-modifying adverb	*in the house*	*fid-dar*
`Ant`	anteriority	simultaneous, anterior	
`CAdv`	comparative adverb	*more than*	*iktar minn*
`CN`	common noun (without determiner)	*red house*	*dar ħamra*
`Card`	cardinal number	*seven*	*sebgħa*
`Cl`	declarative clause, with all tenses	*she walks with you*	*timxi magħkom*
`Comp`	complement of copula, such as AP	*very warm*	*sħun ħafna*
`Conj`	conjunction	*and*	*u*
`Det`	determiner phrase	*those seven*	*dawk is-seba'*
`Digits`	cardinal or ordinal in digits	*1,000/1,000th*	*1,000*
`IAdv`	interrogative adverb	*why*	*għalfejn*
`IComp`	interrogative complement of copula	*where*	*fejn*
`IDet`	interrogative determiner	*how many*	*kemm*
`IP`	interrogative pronoun	*who*	*min*
`Imp`	imperative	*look at this*	*ħares lejn dan*
`Interj`	interjection	*alas*	*sfortunatament*
`N`	common noun	*house*	*dar*
`N2`	relational noun	*brother*	*ħu*
`N3`	three-place relational noun	*distance*	*distanza*
`NP`	noun phrase (subject or object)	*the red house*	*id-dar ħamra*
`Num`	number determining element	*seven*	*sebgħa*
`Numeral`	cardinal or ordinal in words	*five/fifth*	*ħames/ħames*
`Ord`	ordinal number (used in `Det`)	*seventh*	*ħames*
`PConj`	phrase-beginning conjunction	*therefore*	*allura*
`PN`	proper name	*Paris*	*Pariġi*
`Phr`	phrase in a text	*but don't walk*	*imma timxix*
`Pol`	polarity	positive, negative	
`Predet`	predeterminer (prefixed `Quant`)	*all*	*kollha*
`Prep`	preposition, or just case	*in*	*fi*
`Pron`	personal pronoun	*she*	*hi*
`QCl`	question clause, with all tenses	*why does she walk*	*għalfejn timxi*

Continued on next page

`QS`	question	*where did she live*	*fejn għexet*
`Quant`	quantifier ('nucleus' of Det)	*this/these*	*dan/dawn*
`RCl`	relative clause, with all tenses	*in which she lives*	*li fiha tgħix*
`RP`	relative pronoun	*in which*	*li fiha*
`RS`	relative	*in which she lived*	*li fiha għexet*
`S`	declarative sentence	*she lived here*	*għexet hawn*
`SC`	embedded sentence or question	*that he thinks*	*li jaħseb*
`Subj`	subjunction	*if*	*jekk*
`Temp`	temporal and aspectual features	past anterior	
`Tense`	tense	present, past, future	
`Text`	text consisting of several phrases	*He is here. Why?*	*Huwa hawn. Għalfejn?*
`Utt`	sentence, question, word…	*be quiet*	*oqgħod kwiet*
`V`	one-place verb	*sleep*	*raqad*
`V2`	two-place verb	*love*	*ħabb*
`V2A`	verb with NP and AP complement	*paint*	*pitter*
`V2Q`	verb with NP and Q complement	*ask*	*saqsa*
`V2S`	verb with NP and S complement	*answer*	*wieġeb*
`V2V`	verb with NP and V complement	*beg*	*ttallab*
`V3`	three-place verb	*give*	*ta*
`VA`	adjective-complement verb	*become*	*sar*
`VP`	verb phrase	*is very warm*	*huwa sħun ħafna*
`VPSlash`	verb phrase missing complement	*give to John*	*ta lil Ġanni*
`VQ`	question-complement verb	*wonder*	*kkuntempla*
`VS`	sentence-complement verb	*say*	*qal*
`VV`	verb-phrase-complement verb	*want*	*ried*
`Voc`	vocative or "please"	*please*	*jekk jgħoġbok*

A.2 Modules

Figure A.2 shows the module structure of the Maltese resource grammar. This follows the standard module division in the RGL. Tables A.2 and A.3 describe the contents of each module.

Table A.2: Summary of concrete modules in the Resource Grammar Library (for non-functor languages). Adapted from http://www.grammaticalframework.org/lib/doc/gfdoc/sources.html

Concrete module	Contains
`All`	Top-level module including irregular and extra
`Lang`	Combination of grammar and lexicon
`Irreg`	Irregularly inflected words
`Extra`	Language-specific extra constructs not available via the common API
`Grammar`	All grammar rules, but no content lexicon

Continued on next page

`Lexicon`	Content word lexicon (348 entries, common to all languages)
`Dict`	Large-scale monolingual dictionary, typically extracted from an open-source lexicon
`Structural`	Structural word lexicon
`Noun`	Constructors for `NP`, `CN`, `Det`
`Verb`	Constructors for `VP`, `VPSlash`, `Comp`
`Adjective`	Constructors for `A`, `AP`
`Adverb`	Constructors for `Adv`, `AdV`
`Numeral`	Constructors for `Numeral`, `Digits`
`Sentence`	Constructors for `S`, `Cl`, `SC`
`Relative`	Constructors for `RS`, `RCl`, `RP`
`Conjunction`	Coordination rules
`Phrase`	Constructors for `Phr`, `Utt`
`Idiom`	Idiomatic constructions
`Question`	Constructors for `QS`, `QCl`, `IP`
`Cat`	Lincats of all categories

Table A.3: Summary of common resource modules

Resource module	Contains
`Paradigms`	Morphological paradigms for constructing lexical entries
`Morpho`	Low-level morphological operations
`Res`	Basic `oper` and `param` definitions which are used throughout the grammar

A.3 Paradigms

Tables A.4 to A.7 list the paradigms and constants in the Maltese resource grammar, made available through the `ParadigmsMlt` module.

Table A.4: Morphological paradigms: constructors

Function	Type	Explanation
`masculine`	`Gender`	-
`feminine`	`Gender`	-
`singular`	`Number`	-
`plural`	`Number`	-
`form1`	`VDerivedForm`	Binyan I: *daħal*
`form2`	`VDerivedForm`	Binyan II: *daħħal*
`form3`	`VDerivedForm`	Binyan III: *wieġeb*
`form4`	`VDerivedForm`	Binyan IV: *wera*
`form5`	`VDerivedForm`	Binyan V: *ddaħħal*

Continued on next page

`form6`	`VDerivedForm`	Binyan VI: *twieġeb*
`form7`	`VDerivedForm`	Binyan VII: *ndaħal*
`form8`	`VDerivedForm`	Binyan VIII: *ftakar*
`form9`	`VDerivedForm`	Binyan IX: *sfar*
`form10`	`VDerivedForm`	Binyan X: *stieden*
`strong`	`VClass`	Strong tri. verb: *kiteb* ($\sqrt{\text{KTB}}$)
`liquidMedial`	`VClass`	Strong liquid-medial tri. verb: *ħareġ* ($\sqrt{\text{ĦRĠ}}$)
`geminated`	`VClass`	Strong geminated tri. verb: *ħabb* ($\sqrt{\text{ĦBB}}$)
`assimilative`	`VClass`	Weak-initial tri. verb: *wieġeb* ($\sqrt{\text{WĠB}}$)
`hollow`	`VClass`	Weak-medial tri. verb: *ried* ($\sqrt{\text{RJD}}$)
`lacking`	`VClass`	Weak-final tri. verb: *mexa* ($\sqrt{\text{MXJ}}$)
`defective`	`VClass`	Għ-final tri. verb: *qata'* ($\sqrt{\text{QTGĦ}}$)
`quad`	`VClass`	Strong quad. verb: *ħarbat* ($\sqrt{\text{ĦRBT}}$)
`quadWeak`	`VClass`	Weak-final quad. verb: *kanta* ($\sqrt{\text{KNTJ}}$)
`irregular`	`VClass`	Irregular verb: *af* ($\sqrt{\text{''F}}$)
`loan`	`VClass`	Loan verb: *ipparkja* (no root)

Table A.5: Morphological paradigms: nouns

Function	Type	Explanation
`mkN`	`Str -> N`	Noun paradigm 1: Take the singular and infer plural
`mkN`	`Str -> Gender -> N`	Noun paradigm 1: Explicit gender
`mkN`	`Str -> Str -> N`	Noun paradigm 1: Take the singular and explicit plural
`mkN`	`Str -> Str -> Gender -> N`	Noun paradigm 1: Explicit gender
`mkN`	`Str -> Str -> Str -> N`	Noun paradigm 1x: Take singular and both plurals
`mkN`	`Str -> Str -> Str -> Gender -> N`	Noun paradigm 1x: Explicit gender
`mkNColl`	`Str -> N`	Noun paradigm 2c: Collective form only
`mkNColl`	`Str -> Str -> N`	Noun paradigm 2b: Collective and plural
`mkNColl`	`Str -> Str -> Str -> N`	Noun paradigm 2: Singular, collective and plural
`mkNColl`	`Str -> Str -> Str -> Str -> N`	Noun paradigm 2x: Singular, collective and both plurals
`mkNNoPlural`	`Str -> N`	Noun paradigm 3: No plural
`mkNNoPlural`	`Str -> Gender -> N`	Noun paradigm 3: Explicit gender
`mkNDual`	`Str -> N`	Noun paradigm 4: Infer dual, plural and gender from singular

Continued on next page

mkNDual	Str -> Str -> Str -> N	Noun paradigm 4: Singular, dual, plural
mkNDual	Str -> Str -> Str -> Gender -> N	Noun paradigm 4: Explicit gender
mkNDual	Str -> Str -> Str -> Str -> N	Noun paradigm 4x: Singular, dual, both plurals
mkNDual	Str -> Str -> Str -> Str -> Gender -> N	Noun paradigm 4x: Explicit gender
mkPN	Str -> Gender -> Number -> ProperNoun	Proper noun
mkN2	N -> Prep -> N2	-
mkN2	N -> Str -> N2	-
mkN2	N -> N2	use “ta”’
mkN3	Noun -> Prep -> Prep -> N3	-
possN	N -> N	Mark a noun as taking possessive enclitic pronouns: *missieri*, *missierek*...

Table A.6: Morphological paradigms: verbs

Function	Type	Explanation
mkRoot	Root	Null root
mkRoot	Str -> Root	From hyphenated string: `"k-t-b"`
mkRoot	Str -> Str -> Str -> Root	Tri-consonantal root
mkRoot	Str -> Str -> Str -> Str -> Root	Quadri-consonantal root
mkVowels	Vowels	Null vowel sequence
mkVowels	Str -> Vowels	Only single vowel
mkVowels	Str -> Str -> Vowels	Two-vowel sequence
mkV	Str -> V	With no root, automatically treat as loan verb
mkV	Str -> Root -> V	Take an explicit root, implying it is a root & pattern verb
mkV	Str -> Str -> Root -> V	Takes an Imperative of the word for when it behaves less predictably
mkV	VClass -> VDerivedForm -> Root -> Vowels -> Str -> Str -> Str -> Str -> Str -> Str -> Str -> Str -> Str -> Str -> Str -> Str -> Str -> V	All forms: `mkV (Strong Regular) (FormI) (mkRoot "k-t-b") (mkVowels "i" "e") "ktibt" "ktibt" "kiteb" "kitbet" "ktibna" "ktibtu" "kitbu" "nikteb" "tikteb" "jikteb" "tikteb" "niktbu" "tiktbu" "jiktbu" "ikteb" "iktbu"`
mkV_II	Str -> Root -> V	Form II verb: `mkV_II "waqqaf" (mkRoot "w-q-f")`
mkV_II	Str -> Str -> Root -> V	Form II verb with explicit imperative form: `mkV_II "waqqaf" "waqqaf" (mkRoot "w-q-f")`
mkV_III	Str -> Root -> V	Form III verb: `mkV_III "qiegħed" (mkRoot "q-għ-d")`
mkV_V	Str -> Root -> V	Form V verb: `mkV_V "twaqqaf" (mkRoot "w-q-f")`
mkV_VI	Str -> Root -> V	Form VI verb: `mkV_VI "tqiegħed" (mkRoot "q-għ-d")`

Continued on next page

`mkV_VII`	`Str -> Str -> Root -> V`	Form VII verb: `mkV_VII "xeħet" "nxteħet" (mkRoot "x-ħ-t")`
`mkV_VIII`	`Str -> Root -> V`	Form VIII verb: `mkV_VIII "xteħet" (mkRoot "x-ħ-t")`
`mkV_IX`	`Str -> Root -> V`	Form IX verb: `mkV_IX "sfar" (mkRoot "s-f-r")`
`mkV_X`	`Str -> Root -> V`	Form X verb: `mkV_X "stagħġeb" (mkRoot "għ-ġ-b")`
`presPartV`	`Str -> V -> V`	Add the present participle to a verb: *ħiereġ*
`presPartV`	`Str -> Str -> Str -> V -> V`	Add the present participle to a verb: *ħiereġ, ħierġa, ħierġin*
`pastPartV`	`Str -> V -> V`	Add the past participle to a verb: *miktub*
`pastPartV`	`Str -> Str -> Str -> V -> V`	Add the past participle to a verb: *miktub, miktuba, miktubin*
`mkVS`	`V -> VS`	sentence-compl
`mkV3`	`V -> V3`	ditransitive: *ta, _, _*
`mkV3`	`V -> Prep -> Prep -> V3`	two prepositions: *kellem, ma', fuq*
`mkV3`	`V -> Prep -> V3`	one preposition: *ta, _, lil*
`mkV2V`	`V -> Prep -> Prep -> V2V`	*ried, _, lil*

Table A.7: Morphological paradigms: others

Function	Type	Explanation
`mkConj`	`Str -> Conj`	Conjunction: *wieħed tnejn u tlieta*
`mkConj`	`Str -> Str -> Conj`	Conjunction: *wieħed , tnejn u tlieta*
`mkA`	`Str -> A`	Regular adjective with predictable feminine and plural forms: *bravu*
`mkA`	`Str -> Str -> A`	Infer feminine from masculine; no comparative form: *sabiħ, sbieħ*
`mkA`	`Str -> Str -> Str -> A`	Explicit feminine form; no comparative form: *sabiħ, sabiħa, sbieħ*
`mkA`	`Str -> Str -> Str -> Str -> A`	All forms: *sabiħ, sabiħa, sbieħ, isbaħ*
`sameA`	`Str -> A`	Adjective with same forms for masculine, feminine and plural: *blu*
`mkA2`	`A -> Prep -> A2`	-
`mkA2`	`A -> Str -> A2`	-
`mkAS`	`A -> AS`	-
`mkAdv`	`Str -> Adv`	post-verbal adverb: *illum*
`mkAdV`	`Str -> AdV`	preverbal adverb: *dejjem*
`mkAdA`	`Str -> AdA`	adverb modifying adjective: *pjuttost*
`mkAdN`	`Str -> AdN`	adverb modifying numeral: *madwar*

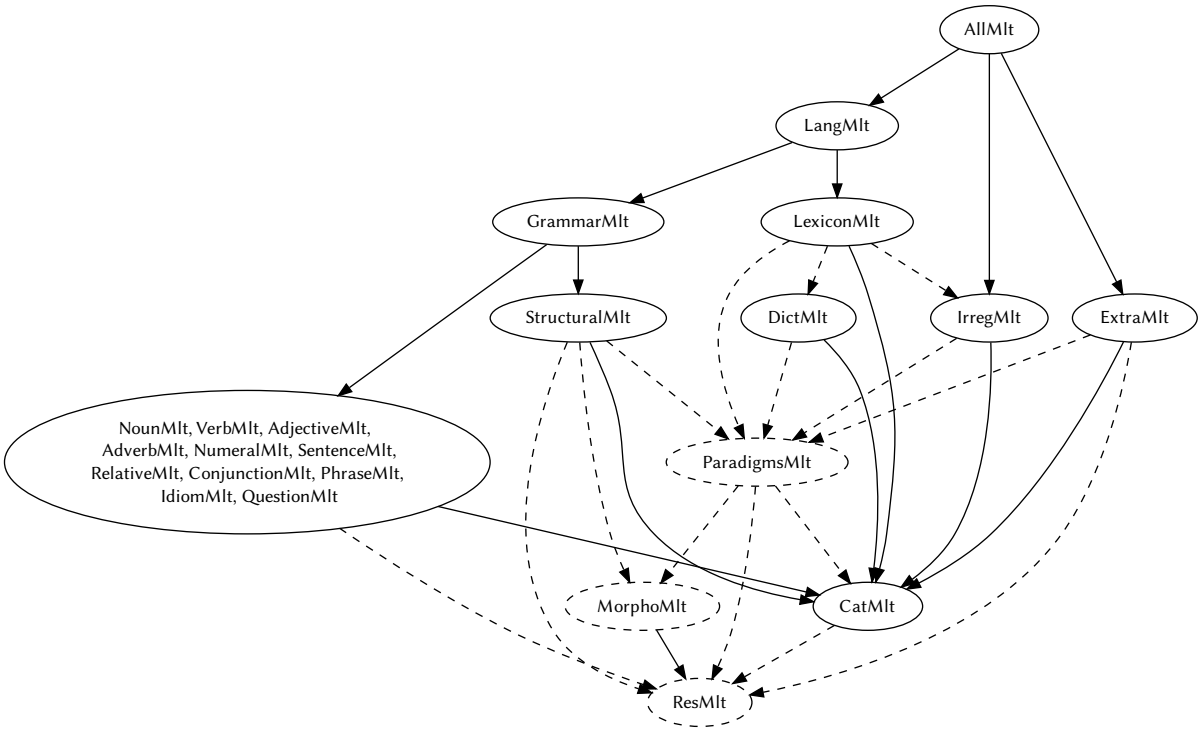


Figure A.2: Module dependency graph of the Maltese resource grammar. Dotted ellipses indicate resource modules, while dotted lines indicate opening (as opposed to module extension). Modules `NounMlt` to `QuestionMlt` are grouped together only for clarity. Abstract modules have been omitted.

Appendix B

Lexicon analyses

B.1 Analysis of noun paradigms

Table B.1 shows an analyses of the noun entries from the common RGL lexicon. For each, we have tabulated all possible forms of the noun assigned a paradigm number, described below. For more information about the different noun paradigms, please refer to section 2.2.3.

1(x)	Singular, Determinate plural (with Indeterminate plural)
2(x)	Singular, Collective, Determinate plural (with Indeterminate plural)
2b(x)	Collective, Determinate plural (with Indeterminate plural)
2c	Collective only
3	Singular only
4(x)	Singular, Dual, Determinate plural (with Indeterminate plural)

B.1.1 Determinate and indeterminate plurals

While nouns many have both plural forms, it seems that in reality there are very few nouns which actually have both simultaneously. Specifically:

- 14 (~7%) have both forms, though many of these arguably sound archaic, e.g. *ġbiel* (for *ġebliet*, 'stones'), *xgħur* (for *xagħariet*, 'hairs'), *għejun* (for *għajnejn*, 'eyes').
- 158 (~86%) have just a determinate plural
- 3 (~1%) have just an indeterminate plural
- 9 (~5%) have neither plural form. This is usually compensated by a collective form (e.g. *baqar*, 'cows'), a dual (e.g. *riġlejn*, 'legs') or simply a singulative (e.g. *plastik*, 'plastic').

Table B.1: Analysis of noun plural forms

Lexicon entry	Paradigm	Singular	Collective	Dual	Det. Plural	Ind. Plural
airplane_N	1	*ajruplan*	-	-	*ajruplani*	-
animal_N	1	*annimal*	-	-	*annimali*	-
apartment_N	1	*appartament*	-	-	*appartamenti*	-
apple_N	2	*tuffieħa*	*tuffieħ*	-	*tuffieħat*	-
art_N	3	*arti*	-	-	-	-
ashes_N	1	*rmied*	-	-	*rmiet*	-
baby_N	1	*tarbija*	-	-	*trabi*	-
back_N	1	*dahar*	-	-	*dhur*	-
bank_N	1	*bank*	-	-	*bankijiet*	-
bark_N	1	*qoxra*	-	-	*qoxriet*	-
beer_N	1	*birra*	-	-	*birer*	-
belly_N	1	*żaqq*	-	-	*żquq*	-
bike_N	1	*rota*	-	-	*roti*	-
bird_N	1	*għasfur*	-	-	*għasafar*	-
blood_N	2b	-	*demm*	-	*dmija*	-
boat_N	1	*dgħajsa*	-	-	*dgħajjes*	-
bone_N	2	*għadma*	*għadam*	-	*għadmiet*	-
book_N	1	*ktieb*	-	-	*kotba*	-
boot_N	2	*żarbuna*	*żarbun*	-	*żraben*	-
boss_N	1	*mgħallem*	-	-	*mgħallmin*	-
boy_N	1	*tifel*	-	-	*tfal*	-
bread_N	2	*ħobża*	*ħobż*	-	*ħobżiet*	-
breast_N	1	*sider*	-	-	*sdur*	-
brother_N2	1	*ħu*	-	-	*aħwa*	-
butter_N	2b	-	*butir*	-	*butirijiet*	-
camera_N	1	*kamera*	-	-	*kameras*	-
cap_N	1	*kappell*	-	-	*kpiepel*	-
car_N	1	*karozza*	-	-	*karozzi*	-
carpet_N	1	*tapit*	-	-	*twapet*	-
cat_N	1	*qattus*	-	-	*qtates*	-
ceiling_N	1	*saqaf*	-	-	*soqfa*	-
chair_N	1	*siġġu*	-	-	*siġġijiet*	-
cheese_N	2	*ġobna*	*ġobon*	-	*ġobniet*	-
child_N	1	*tifel/tifla*	-	-	*tfal*	-
church_N	1	*knisja*	-	-	*knejjes*	-
city_N	1	*belt*	-	-	*bliet*	-
cloud_N	2	*sħaba*	*sħab*	-	*sħabiet*	-
coat_N	1	*kowt*	-	-	*kowtijiet*	-

Continued on next page

computer_N	1	*kompjuter*	-	-	*kompjuters*	-
country_N	1	*pajjiż*	-	-	*pajjiżi*	-
cousin_N	1	*kuġin*	-	-	*kuġini*	-
cow_N	2	*baqra*	*baqar*	-	*baqriet*	-
day_N	1	*ġurnata*	-	-	*ġranet*	-
distance_N3	1	*distanza*	-	-	*distanzi*	-
doctor_N	1	*tabib*	-	-	*tobba*	-
dog_N	1	*kelb*	-	-	*klieb*	-
door_N	1	*bieb*	-	-	*bibien*	-
dust_N	2	*traba*	*trab*	-	*trabiet*	-
ear_N	4	*widna*	-	*widnejn*	*widniet*	-
earth_N	1	*art*	-	-	*artijiet*	-
egg_N	2	*bajda*	*bajd*	-	*bajdiet*	-
enemy_N	1	*għadu*	-	-	*għedewwa*	-
eye_N	4x	*għajn*	-	*għajnejn*	*għajnejn*	*għejun*
factory_N	1	*fabbrika*	-	-	*fabbriki*	-
fat_N	2x	*xaħma*	*xaħam*	-	*xaħmiet*	*xaħmijiet*
father_N2	1	*missier*	-	-	*missirijiet*	-
feather_N	2	*rixa*	*rix*	-	*rixiet*	-
fingernail_N	4	*difer*	-	*difrejn*	*dwiefer*	-
fire_N	1	*nar*	-	-	*nirien*	-
fish_N	2	*ħuta*	*ħut*	-	*ħutiet*	-
floor_N	1	*art*	-	-	*artijiet*	-
flower_N	1	*fjura*	-	-	*fjuri*	-
fog_N	2c	-	*ċpar*	-	-	-
foot_N	4	*sieq*	-	*saqajn*	*saqajn*	-
forest_N	1	*foresta*	-	-	*foresti*	-
fridge_N	1	*friġġ*	-	-	*friġġijiet*	-
friend_N	1	*ħabib*	-	-	*ħbieb*	-
fruit_N	2x	*frotta*	*frott*	-	*frottiet*	*frottijiet*
garden_N	1	*ġnien*	-	-	*ġonna*	-
girl_N	1	*tifla*	-	-	*tfal*	-
glove_N	1	*ingwanta*	-	-	*ingwanti*	-
gold_N	2b	-	*deheb*	-	*dehbijiet*	-
grammar_N	1	*grammatika*	-	-	*grammatiki*	-
grass_N	2	*ħaxixa*	*ħaxix*	-	*ħxejjex*	-
guts_N	1x	*musrana*	-	-	*musraniet*	*mɪsaren*
hair_N	2x	*xagħara*	*xagħar*	-	*xagħariet*	*xgħur*
hand_N	4	*id*	-	*idejn*	*idejn*	-
harbour_N	1	*port*	-	-	*portijiet*	-
hat_N	1	*kappell*	-	-	*kpiepel*	-

Continued on next page

head_N	1	*ras*	-	-	*rjus*	-
heart_N	1	*qalb*	-	-	*qlub*	-
hill_N	1	*għolja*	-	-	*għoljiet*	-
horn_N	1	*ħorn*	-	-	*ħornijiet*	-
horse_N	1	*żiemel*	-	-	*żwiemel*	-
house_N	1	*dar*	-	-	*djar*	-
husband_N	1	*raġel*	-	-	*rġiel*	-
ice_N	1	*silġ*	-	-	*silġiet*	-
industry_N	1	*industrija*	-	-	*industriji*	-
iron_N	2x	*ħadida*	*ħadid*	-	*ħadidiet*	*ħdejjed*
king_N	1	*re*	-	-	*rejjiet*	-
knee_N	4	*rkoppa*	-	*rkopptejn*	*rkoppiet*	-
lake_N	1	*għadira*	-	-	*għadajjar*	-
lamp_N	1	*lampa*	-	-	*lampi*	-
language_N	1	*lingwa*	-	-	*lingwi*	-
leaf_N	4	*werqa*	-	*werqtejn*	*werqiet*	-
leather_N	1	*ġilda*	-	-	*ġildiet*	-
leg_N	4	*riġel*	-	*riġlejn*	*riġlejn*	-
liver_N	1	*fwied*	-	-	*ifdwa*	-
louse_N	1	*qamla*	-	-	*qamliet*	-
love_N	1	*mħabba*	-	-	*mħabbiet*	-
man_N	1	*raġel*	-	-	*rġiel*	-
meat_N	2x	*laħma*	*laħam*	-	*laħmiet*	*laħmijiet*
milk_N	2bx	-	*ħalib*	-	*ħalibijiet*	*ħlejjeb*
moon_N	1	*qamar*	-	-	*oqmra*	-
mother_N2	1	*omm*	-	-	*ommijiet*	-
mountain_N	1	*muntanja*	-	-	*muntanji*	-
mouth_N	1	*ħalq*	-	-	*ħluq*	-
music_N	1	*mużika*	-	-	*mużiki*	-
name_N	1	*isem*	-	-	*ismijiet*	-
neck_N	1	*għonq*	-	-	*għenuq*	-
newspaper_N	1	*gazzetta*	-	-	*gazzetti*	-
night_N	1	*lejl*	-	-	*ljieli*	-
nose_N	1	*mnieħer*	-	-	*mniħrijiet*	-
number_N	1	*numru*	-	-	*numrui*	-
oil_N	1	*żejt*	-	-	*żjut*	-
paper_N	1	*karta*	-	-	*karti*	-
peace_N	1	*paċi*	-	-	*paċijiet*	-
pen_N	1	*pinna*	-	-	*pinen*	-
person_N	2b	-	*persuna*	-	*persuni*	-
planet_N	1	*pjaneta*	-	-	*pjaneti*	-

Continued on next page

plastic_N	3	*plastik*	-	-	-	-
policeman_N	3	*pulizija*	-	-	-	-
priest_N	1	*qassis*	-	-	*qassisin*	-
queen_N	1	*reġina*	-	-	*rġejjen*	-
question_N	1	*mistoqsija*	-	-	*mistoqsijiet*	-
radio_N	1	*radju*	-	-	*radjijiet*	-
rain_N	3	*xita*	-	-	-	-
reason_N	1	*raġun*	-	-	*raġunijiet*	-
religion_N	1	*reliġjon*	-	-	*reliġjonijiet*	-
restaurant_N	1	*restorant*	-	-	*restoranti*	-
river_N	1	*xmara*	-	-	*xmajjar*	-
road_N	1x	*triq*	-	-	*triqat*	*toroq*
rock_N	2	*blata*	*blat*	-	*blatiet*	-
roof_N	1	*saqaf*	-	-	*soqfa*	-
root_N	1	*qħerq*	-	-	*qħeruq*	-
rope_N	1	*ħabel*	-	-	*ħbula*	-
rubber_N	1	*gomma*	-	-	*gomom*	-
rule_N	1	*regola*	-	-	*regoli*	-
salt_N	1	*melħ*	-	-	*melħiet*	-
sand_N	2	*ramla*	*ramel*	-	*ramliet*	-
school_N	1	*skola*	-	-	*skejjel*	-
science_N	1	*xjenza*	-	-	*xjenzi*	-
sea_N	4	*baħar*	-	*baħrejn*	*ibħra*	-
seed_N	1	*żerriegħa*	-	-	*żerrigħat*	-
sheep_N	2	*nagħġa*	*ngħaġ*	-	*nagħġiet*	-
ship_N	1	*vapur*	-	-	*vapuri*	-
shirt_N	1	*qmis*	-	-	*qomos*	-
shoe_N	1	*żarbun*	-	-	*żraben*	-
shop_N	1	*ħanut*	-	-	*ħwienet*	-
silver_N	1	*fidda*	-	-	*fided*	-
sister_N	1	*oħt*	-	-	*aħwa*	-
skin_N	1	*ġilda*	-	-	*ġildiet*	-
sky_N	1	*sema*	-	-	*smewwiet*	-
smoke_N	1	*duħħan*	-	-	*dħaħen*	-
snake_N	1	*serp*	-	-	*sriep*	-
snow_N	2c	-	*borra*	-	-	-
sock_N	1	*kalzetta*	-	-	*kalzetti*	-
song_N	1	*kanzunetta*	-	-	*kanzunetti*	-
star_N	1	*stilla*	-	-	*stilel*	-
steel_N	3	*azzar*	-	-	-	-
stick_N	1	*lasta*	-	-	*lasti*	-

Continued on next page

stone_N	2x	*ġebla*	*ġebel*	-	*ġebliet*	*ġbiel*
stove_N	1	*kuker*	-	-	*kukers*	-
student_N	1	*student*	-	-	*studenti*	-
sun_N	1	*xemx*	-	-	*xmux*	-
table_N	1	*mejda*	-	-	*mwejjed*	-
tail_N	1	*denb*	-	-	*dnieb*	-
teacher_N	1	*għalliem*	-	-	*għalliema*	-
television_N	1	*televixin*	-	-	*televixins*	-
tongue_N	1	*lsien*	-	-	*ilsna*	-
tooth_N	1x	*sinna*	-	-	*sinniet*	*snien*
train_N	1	*ferrovija*	-	-	*ferroviji*	-
tree_N	2	*siġra*	*siġar*	-	*siġriet*	-
university_N	1	*università*	-	-	*universitàjiet*	-
village_N	1	*raħal*	-	-	*rħula*	-
war_N	1	*gwerra*	-	-	*gwerrer*	-
water_N	1	*ilma*	-	-	*ilmijiet*	-
wife_N	1	*mara*	-	-	*nisa*	-
wind_N	1	*riħ*	-	-	*rjieħ*	-
window_N	1	*tieqa*	-	-	*twieqi*	-
wine_N	2b	-	*nbid*	-	*nbejjed*	-
wing_N	1	*ġewnaħ*	-	-	*ġwienaħ*	-
woman_N	1	*mara*	-	-	*nisa*	-
wood_N	1	*injam*	-	-	*injamiet*	-
worm_N	2	*dudu*	*dud*	-	*dudiet*	-
year_N	4	*sena*	-	*sentejn*	*snin*	-

B.2 Analysis of verb participles

Another analysis carried out on the RGL lexicon module is that of verb participles. As can be seen in table B.2, many verbs have a past participle but very few have a present one.

Table B.2: Analysis of verb participles

Lexicon entry	Perf. P3 Sg Masc	Present Participle	Past Participle
add_V3	*żied*	-	*miżjud*
answer_V2S	*wieġeb*	-	*mwieġeb*
ask_V2Q	*saqsa*	-	*mistoqsi*
become_VA	*sar*	-	*misjur*
beg_V2V	*ttallab*	-	*mitlub*
bite_V2	*gidem*	-	*migdum*
blow_V	*nefaħ*	-	*minfuħ*
break_V2	*kiser*	-	*miksur*

Continued on next page

breathe_V	*respira*	-	-
burn_V	*ħaraq*	-	*maħruq*
buy_V2	*xtara*	-	*mixtri*
close_V2	*għalaq*	-	*magħluq*
come_V	*ġie*	*ġej*	*miġjub*
count_V2	*għadd*	-	*mgħadd*
cut_V2	*qata′*	-	*maqtugħ*
die_V	*miet*	-	*mejjet*
dig_V	*ħaffer*	-	*mħaffer*
do_V2	*għamel*	-	*magħmul*
drink_V2	*xorob*	-	*mixrub*
eat_V2	*kiel*	-	*mikul*
fall_V	*waqa′*	-	*mwaqqa′*
fear_V2	*beża′*	-	*mbeżża′*
fear_VS	*beża′*	-	*mbeżża′*
fight_V2	*ġġieled*	-	*miġġieled*
find_V2	*sab*	-	*misjub*
float_V	*għam*	-	-
flow_V	*għadda*	-	*mgħoddi*
fly_V	*tar*	-	*mtajjar*
forget_V2	*nesa*	-	*minsi*
freeze_V	*ffriza*	-	*ffrizat*
give_V3	*ta*	-	*mogħti*
go_V	*mar*	*sejjer*	-
hate_V2	*bagħad*	-	*mibgħud*
hear_V2	*sema′*	-	*mismugħ*
hit_V2	*laqat*	-	*milqut*
hold_V2	*żamm*	-	*miżjum*
hope_VS	*xtaq*	-	*mixtieq*
hunt_V2	*kaċċaċ*	-	*kkaċċjat*
jump_V	*qabeż*	-	*maqbuż*
kill_V2	*qatel*	-	*maqtul*
know_V2	*af*	-	-
know_VQ	*af*	-	-
know_VS	*af*	-	-
laugh_V	*daħak*	-	-
learn_V2	*tgħallem*	-	-
leave_V2	*telaq*	-	*mitluq*
lie_V	*mtedd*	-	*mimdud*
like_V2	*għoġob*	-	-
listen_V2	*sema′*	-	*mismugħ*

Continued on next page

live_V	*għex*	-	-
lose_V2	*tilef*	-	*mitluf*
love_V2	*ħabb*	-	*maħbub*
open_V2	*fetaħ*	-	*miftuħ*
paint_V2A	*pitter*	-	*mpitter*
play_V	*daqq*	-	-
play_V2	*lagħab*	-	-
pull_V2	*ġibed*	-	*miġbud*
push_V2	*mbotta*	-	*mbuttat*
put_V2	*qiegħed*	-	*mqiegħed*
rain_V0	*xita*	-	-
read_V2	*qara*	-	*moqri*
rub_V2	*għorok*	-	-
run_V	*ġera*	-	*miġri*
say_VS	*qal*	-	-
scratch_V2	*barax*	-	*mibrux*
see_V2	*ra*	-	*muri*
seek_V2	*fittex*	-	*mfittex*
sell_V3	*biegħ*	-	-
send_V3	*bagħat*	-	*mibgħut*
sew_V	*ħat*	-	*meħjut*
sing_V	*kanta*	-	*kantat*
sit_V	*poġġa*	-	*poġġut*
sleep_V	*raqad*	*rieqed*	*mraqqad*
smell_V	*xamm*	-	*mixmum*
speak_V2	*kellem*	-	*mitkellem*
spit_V	*beżaq*	-	*mibżuq*
split_V2	*qasam*	-	*maqsum*
squeeze_V2	*għasar*	-	*mgħasur*
stab_V2	*mewwes*	-	*mmewwes*
stand_V	*qagħad*	-	*mqiegħed*
stop_V	*waqaf*	*wieqaf*	*mwaqqaf*
suck_V2	*rada'*	-	*mradda'*
swell_V	*ntefaħ*	-	*minfuħ*
swim_V	*għam*	-	-
switch8off_V2	*tefa*	-	*mitfi*
switch8on_V2	*xegħel*	-	*mixgħul*
talk_V3	*kellem*	-	*mitkellem*
teach_V2	*għallem*	-	*mgħallem*
think_V	*ħaseb*	*ħosbien*	*maħsub*
throw_V2	*waddab*	-	*mwaddab*

Continued on next page

tie_V2	*qafel*	-	*maqful*
travel_V	*vvjaġġa*	-	*vvjaġġat*
turn_V	*dar*	-	*mdawwar*
understand_V2	*fehem*	-	*mifhum*
vomit_V	*qala'*	-	*maqlugħ*
wait_V2	*stenna*	-	*mistenni*
walk_V	*mexa*	*miexi*	*mmexxi*
wash_V2	*ħasel*	-	*maħsul*
watch_V2	*ra*	-	-
win_V2	*rebaħ*	-	*mirbuħ*
wipe_V2	*mesaħ*	-	*mimsuħ*
wonder_VQ	*kkuntempla*	-	*kkuntemplat*
write_V2	*kiteb*	-	*miktub*

B.3 Vowel changes

As in many languages, affixation in Maltese often causes vowel changes in the word stem. The sections below cover some common ambiguous cases of this.

B.3.1 Object suffixes

Consider the inflected verb form *rajna* ('we saw') and the pronominal suffix *-k* indicating P2 sg. as a direct object. According to (Brother Henry F.S.C., 1980, p. 166), whenever a verb ending in *-a* has an enclitic pronoun suffixed to it, the joining vowel changes to an *ie*. Thus *rajna* + *-k* → *rajniek* ('we saw you'). This rule is clearly established, despite the fact that based on how it sounds as a native speaker a shorter-vowel spelling *rajnik* seems more natural.

Searching for both versions in the Maltese corpus, in fact, reveals that both forms appear equally as often as each other. In general, looking at the frequency counts for tokens ending in *-jniek* and *-jnik*, the former occurs 17 times and the latter 18 times.

A identical situation exists with enclitic pronouns for indirect objects too. Thus *kantajna* + *-ilek* → *kantajnielek* ('we sang for you'), even though *kantajnilek* sounds like a more accurate transcription of the spoken form. The corpus results are again split, reporting 11 occurrences of tokens ending in *jnielek*, and 10 for *jnilek*. (Borg & Azzopardi-Alexander, 1997) claims the former is correct, with an *ie*.

When combining both direct and indirect pronominal suffixes, one would expect the same rule as above to apply. This would produce the forms *rajniehulek* ('we saw it for you') and *kantajniehulek* ('we sang it for you'). The corpus, however, contains exactly zero tokens which end with *iehulek*, and a significant 92 which finish with *ihulek*. So in this case, the corpus results seem to contradict the rule. Some personal communication on the 'Kelmet il-Malti' Facebook group[1] seems to indicate that the rule cited above no longer applies, and the more natural

[1] https://www.facebook.com/groups/246657308743181/, accessed 2012-08-29

principle of vowel length seems to dictate the orthography. So we instead get *kantajnihulek* and *ftaħnihulek* as correct forms.

B.3.2 Vowel length and negation

Consider the verbs *waqaf* ('he stopped'), *kiel* ('he ate'), and *ħa* ('he took'). #Note that the latter two are irregular, however I think they are still valid for the point I want to make. Their imperfect forms all consist of a stem containing the long vowel *ie*: *jieqaf*, *jiekol*, and *jieħu* respectively. Negation of the Maltese verb, which involves the suffixation of *-x*, tends to move emphasis of towards the end of the word and means that any long vowel *ie* must change into a shorter *i* or *e* (Azzopardi, 2007, p. 92). This yields the forms *jiqafx*, *jikolx* and *jiħux* for P3 Sg Masc and *jiqfu*, *jiklux* and *jiħdux* for P3 Pl.

A small frequency analysis in the corpus highlights an interesting pattern. Table B.3 shows the frequency of each variant spelling as a percentage of the number of occurrences of the positive form (for which there is no variation in spelling).

Table B.3: Corpus study of vowel length changes under negation

P3 Sg Masc Pos.	**Neg. *ie***	**Neg. *i***	**P3 Pl Pos.**	**Neg. *ie***	**Neg. *i***
jieqaf	3.76%	4.66%	*jieqfu*	3.45%	3.18%
jiekol	1.44%	1.83%	*jieklu*	1.48%	1.28%
jieħu	0.48%	0.73%	*jieħdu*	0.41%	0.42%

What there numbers show is that when considering the singular negative, the version without the long *ie* vowel is more common in all cases. For example, *jikolx* is more common than *jiekolx* — as the rule dictates. For plural negatives however, it's almost the complete opposite. Put simply, *jieklux* is slightly more frequent than *jiklux*, although admittedly the difference in frequency is less pronounced: 7% in the plural compared to 12% in the singular, for the given example.

In conclusion, it seems that the rules in grammar books do not always agree with the frequency results from the corpus. This is perhaps not surprising, but it does go to show that statistical majority in the corpus does not necessarily aligned with correctness.

Appendix C

Inflectional forms of a Maltese verb

Table C.1 shows the full table of inflectional forms of the verb *fetaħ* ('he opened'). For brevity, we refrain from including glosses in every table row. The meaning of each of the object clitic combinations can be summarised in table 2.6.

Note that we are counting forms with suffixed pronouns as individual forms; in the grammar however they are treated as separate clitics which are bound together with verb stems. Dashes in the right-most column indicate combinations which do not exist for semantic reasons (see Borg & Azzopardi-Alexander (1997, p. 360)). Specifically, in the case of P1 and P2 the object pronoun (both direct and indirect) cannot be of the same person as the subject. Such constructions are technically imaginable though, and the resource grammar does not prevent these forms.

Table C.1: Full inflection table for the verb *fetaħ* ('he opened')

Tense	Subject	Direct obj.	Indirect obj.	Positive	Negative
Perf.	P1 Sg	-	-	*ftaħt*	*ftaħtx*
Perf.	P1 Sg	P1 Sg	-	-	-
Perf.	P1 Sg	P1 Pl	-	-	-
Perf.	P1 Sg	P2 Sg	-	*ftaħtek*	*ftaħtekx*
Perf.	P1 Sg	P2 Pl	-	*ftaħtkom*	*ftaħtkomx*
Perf.	P1 Sg	P3 Sg Masc	-	*ftaħtu*	*ftaħtux*
Perf.	P1 Sg	P3 Sg Fem	-	*ftaħtha*	*ftaħthiex*
Perf.	P1 Sg	P3 Pl	-	*ftaħthom*	*ftaħthomx*
Perf.	P1 Sg	-	P1 Sg	-	-
Perf.	P1 Sg	-	P1 Pl	-	-
Perf.	P1 Sg	-	P2 Sg	*ftaħtlek*	*ftaħtlekx*
Perf.	P1 Sg	-	P2 Pl	*ftaħtilkom*	*ftaħtilkomx*
Perf.	P1 Sg	-	P3 Sg Masc	*ftaħtlu*	*ftaħtlux*
Perf.	P1 Sg	-	P3 Sg Fem	*ftaħtilha*	*ftaħtilhiex*

Continued on next page

Perf.	P1 Sg	-	P3 Pl	*ftaħtilhom*	*ftaħtilhomx*
Perf.	P1 Sg	P3 Sg Masc	P1 Sg	-	-
Perf.	P1 Sg	P3 Sg Masc	P1 Pl	-	-
Perf.	P1 Sg	P3 Sg Masc	P2 Sg	*ftaħthulek*	*ftaħthulekx*
Perf.	P1 Sg	P3 Sg Masc	P2 Pl	*ftaħthulkom*	*ftaħthulkomx*
Perf.	P1 Sg	P3 Sg Masc	P3 Sg Masc	*ftaħthulu*	*ftaħthulux*
Perf.	P1 Sg	P3 Sg Masc	P3 Sg Fem	*ftaħthulha*	*ftaħthulhiex*
Perf.	P1 Sg	P3 Sg Masc	P3 Pl	*ftaħthulhom*	*ftaħthulhomx*
Perf.	P1 Sg	P3 Sg Fem	P1 Sg	-	-
Perf.	P1 Sg	P3 Sg Fem	P1 Pl	-	-
Perf.	P1 Sg	P3 Sg Fem	P2 Sg	*ftaħthielek*	*ftaħthielekx*
Perf.	P1 Sg	P3 Sg Fem	P2 Pl	*ftaħthielkom*	*ftaħthielkomx*
Perf.	P1 Sg	P3 Sg Fem	P3 Sg Masc	*ftaħthielu*	*ftaħthielux*
Perf.	P1 Sg	P3 Sg Fem	P3 Sg Fem	*ftaħthielha*	*ftaħthilhiex*
Perf.	P1 Sg	P3 Sg Fem	P3 Pl	*ftaħthielhom*	*ftaħthielhomx*
Perf.	P1 Sg	P3 Pl	P1 Sg	-	-
Perf.	P1 Sg	P3 Pl	P1 Pl	-	-
Perf.	P1 Sg	P3 Pl	P2 Sg	*ftaħthomlok*	*ftaħthomlokx*
Perf.	P1 Sg	P3 Pl	P2 Pl	*ftaħthomlkom*	*ftaħthomlkomx*
Perf.	P1 Sg	P3 Pl	P3 Sg Masc	*ftaħthomlu*	*ftaħthomlux*
Perf.	P1 Sg	P3 Pl	P3 Sg Fem	*ftaħthomlha*	*ftaħthomlhiex*
Perf.	P1 Sg	P3 Pl	P3 Pl	*ftaħthomlhom*	*ftaħthomlhomx*
Perf.	P1 Pl	-	-	*ftaħna*	*ftaħniex*
Perf.	P1 Pl	P1 Sg	-	-	-
Perf.	P1 Pl	P1 Pl	-	-	-
Perf.	P1 Pl	P2 Sg	-	*ftaħniek*	*ftaħniekx*
Perf.	P1 Pl	P2 Pl	-	*ftaħniekom*	*ftaħniekomx*
Perf.	P1 Pl	P3 Sg Masc	-	*ftaħnieh*	*ftaħnihx*
Perf.	P1 Pl	P3 Sg Fem	-	*ftaħnieha*	*ftaħnihiex*
Perf.	P1 Pl	P3 Pl	-	*ftaħniehom*	*ftaħniehomx*
Perf.	P1 Pl	-	P1 Sg	-	-
Perf.	P1 Pl	-	P1 Pl	-	-
Perf.	P1 Pl	-	P2 Sg	*ftaħnielek*	*ftaħnielekx*
Perf.	P1 Pl	-	P2 Pl	*ftaħnielkom*	*ftaħnielkomx*
Perf.	P1 Pl	-	P3 Sg Masc	*ftaħnielu*	*ftaħnielux*
Perf.	P1 Pl	-	P3 Sg Fem	*ftaħnielha*	*ftaħnilhiex*
Perf.	P1 Pl	-	P3 Pl	*ftaħnielhom*	*ftaħnielhomx*
Perf.	P1 Pl	P3 Sg Masc	P1 Sg	-	-

Continued on next page

Perf.	P1 Pl	P3 Sg Masc	P1 Pl	-	-
Perf.	P1 Pl	P3 Sg Masc	P2 Sg	*ftaħniehulek*	*ftaħniehulekx*
Perf.	P1 Pl	P3 Sg Masc	P2 Pl	*ftaħniehulkom*	*ftaħniehulkomx*
Perf.	P1 Pl	P3 Sg Masc	P3 Sg Masc	*ftaħniehulu*	*ftaħniehulux*
Perf.	P1 Pl	P3 Sg Masc	P3 Sg Fem	*ftaħniehulha*	*ftaħnihulhiex*
Perf.	P1 Pl	P3 Sg Masc	P3 Pl	*ftaħniehulhom*	*ftaħniehulhomx*
Perf.	P1 Pl	P3 Sg Fem	P1 Sg	-	-
Perf.	P1 Pl	P3 Sg Fem	P1 Pl	-	-
Perf.	P1 Pl	P3 Sg Fem	P2 Sg	*ftaħnihielek*	*ftaħnihielekx*
Perf.	P1 Pl	P3 Sg Fem	P2 Pl	*ftaħnihielkom*	*ftaħnihielkomx*
Perf.	P1 Pl	P3 Sg Fem	P3 Sg Masc	*ftaħnihielu*	*ftaħnihielux*
Perf.	P1 Pl	P3 Sg Fem	P3 Sg Fem	*ftaħnihielha*	*ftaħnihilhiex*
Perf.	P1 Pl	P3 Sg Fem	P3 Pl	*ftaħnihielhom*	*ftaħnihielhomx*
Perf.	P1 Pl	P3 Pl	P1 Sg	-	-
Perf.	P1 Pl	P3 Pl	P1 Pl	-	-
Perf.	P1 Pl	P3 Pl	P2 Sg	*ftaħniehomlok*	*ftaħniehomlokx*
Perf.	P1 Pl	P3 Pl	P2 Pl	*ftaħniehomlkom*	*ftaħniehomlkomx*
Perf.	P1 Pl	P3 Pl	P3 Sg Masc	*ftaħniehomlu*	*ftaħniehomlux*
Perf.	P1 Pl	P3 Pl	P3 Sg Fem	*ftaħniehomlha*	*ftaħnihomlhiex*
Perf.	P1 Pl	P3 Pl	P3 Pl	*ftaħniehomlhom*	*ftaħniehomlhomx*
Perf.	P2 Sg	-	-	*ftaħt*	*ftaħtx*
Perf.	P2 Sg	P1 Sg	-	*ftaħtni*	*ftaħtnix*
Perf.	P2 Sg	P1 Pl	-	*ftaħtna*	*ftaħtniex*
Perf.	P2 Sg	P2 Sg	-	-	-
Perf.	P2 Sg	P2 Pl	-	-	-
Perf.	P2 Sg	P3 Sg Masc	-	*ftaħtu*	*ftaħtux*
Perf.	P2 Sg	P3 Sg Fem	-	*ftaħtha*	*ftaħthiex*
Perf.	P2 Sg	P3 Pl	-	*ftaħthom*	*ftaħthomx*
Perf.	P2 Sg	-	P1 Sg	*ftaħtli*	*ftaħtlix*
Perf.	P2 Sg	-	P1 Pl	*ftaħtilna*	*ftaħtilniex*
Perf.	P2 Sg	-	P2 Sg	-	-
Perf.	P2 Sg	-	P2 Pl	-	-
Perf.	P2 Sg	-	P3 Sg Masc	*ftaħtlu*	*ftaħtlux*
Perf.	P2 Sg	-	P3 Sg Fem	*ftaħtilha*	*ftaħtilhiex*
Perf.	P2 Sg	-	P3 Pl	*ftaħtilhom*	*ftaħtilhomx*
Perf.	P2 Sg	P3 Sg Masc	P1 Sg	*ftaħthuli*	*ftaħthulix*
Perf.	P2 Sg	P3 Sg Masc	P1 Pl	*ftaħthulna*	*ftaħthulniex*
Perf.	P2 Sg	P3 Sg Masc	P2 Sg	-	-

Continued on next page

Perf.	P2 Sg	P3 Sg Masc	P2 Pl	-	-
Perf.	P2 Sg	P3 Sg Masc	P3 Sg Masc	*ftaħthulu*	*ftaħthulux*
Perf.	P2 Sg	P3 Sg Masc	P3 Sg Fem	*ftaħthulha*	*ftaħthulhiex*
Perf.	P2 Sg	P3 Sg Masc	P3 Pl	*ftaħthulhom*	*ftaħthulhomx*
Perf.	P2 Sg	P3 Sg Fem	P1 Sg	*ftaħthieli*	*ftaħthielix*
Perf.	P2 Sg	P3 Sg Fem	P1 Pl	*ftaħthielna*	*ftaħthilniex*
Perf.	P2 Sg	P3 Sg Fem	P2 Sg	-	-
Perf.	P2 Sg	P3 Sg Fem	P2 Pl	-	-
Perf.	P2 Sg	P3 Sg Fem	P3 Sg Masc	*ftaħthielu*	*ftaħthielux*
Perf.	P2 Sg	P3 Sg Fem	P3 Sg Fem	*ftaħthielha*	*ftaħthilhiex*
Perf.	P2 Sg	P3 Sg Fem	P3 Pl	*ftaħthielhom*	*ftaħthielhomx*
Perf.	P2 Sg	P3 Pl	P1 Sg	*ftaħthomli*	*ftaħthomlix*
Perf.	P2 Sg	P3 Pl	P1 Pl	*ftaħthomlna*	*ftaħthomlniex*
Perf.	P2 Sg	P3 Pl	P2 Sg	-	-
Perf.	P2 Sg	P3 Pl	P2 Pl	-	-
Perf.	P2 Sg	P3 Pl	P3 Sg Masc	*ftaħthomlu*	*ftaħthomlux*
Perf.	P2 Sg	P3 Pl	P3 Sg Fem	*ftaħthomlha*	*ftaħthomlhiex*
Perf.	P2 Sg	P3 Pl	P3 Pl	*ftaħthomlhom*	*ftaħthomlhomx*
Perf.	P2 Pl	-	-	*ftaħtu*	*ftaħtux*
Perf.	P2 Pl	P1 Sg	-	*ftaħtuni*	*ftaħtunix*
Perf.	P2 Pl	P1 Pl	-	*ftaħtuna*	*ftaħtuniex*
Perf.	P2 Pl	P2 Sg	-	-	-
Perf.	P2 Pl	P2 Pl	-	-	-
Perf.	P2 Pl	P3 Sg Masc	-	*ftaħtuh*	*ftaħtuhx*
Perf.	P2 Pl	P3 Sg Fem	-	*ftaħtuha*	*ftaħtuhiex*
Perf.	P2 Pl	P3 Pl	-	*ftaħtuhom*	*ftaħtuhomx*
Perf.	P2 Pl	-	P1 Sg	*ftaħtuli*	*ftaħtulix*
Perf.	P2 Pl	-	P1 Pl	*ftaħtula*	*ftaħtuliex*
Perf.	P2 Pl	-	P2 Sg	-	-
Perf.	P2 Pl	-	P2 Pl	-	-
Perf.	P2 Pl	-	P3 Sg Masc	*ftaħtulu*	*ftaħtulux*
Perf.	P2 Pl	-	P3 Sg Fem	*ftaħtulha*	*ftaħtulhiex*
Perf.	P2 Pl	-	P3 Pl	*ftaħtulhom*	*ftaħtulhomx*
Perf.	P2 Pl	P3 Sg Masc	P1 Sg	*ftaħtuhuli*	*ftaħtuhulix*
Perf.	P2 Pl	P3 Sg Masc	P1 Pl	*ftaħtuhulna*	*ftaħtuhulniex*
Perf.	P2 Pl	P3 Sg Masc	P2 Sg	-	-
Perf.	P2 Pl	P3 Sg Masc	P2 Pl	-	-
Perf.	P2 Pl	P3 Sg Masc	P3 Sg Masc	*ftaħtuhulu*	*ftaħtuhulux*

Continued on next page

Perf.	P2 Pl	P3 Sg Masc	P3 Sg Fem	*ftaħtuhulha*	*ftaħtuhulhiex*
Perf.	P2 Pl	P3 Sg Masc	P3 Pl	*ftaħtuhulhom*	*ftaħtuhulhomx*
Perf.	P2 Pl	P3 Sg Fem	P1 Sg	*ftaħtuhieli*	*ftaħtuhielix*
Perf.	P2 Pl	P3 Sg Fem	P1 Pl	*ftaħtuhielna*	*ftaħtuhilniex*
Perf.	P2 Pl	P3 Sg Fem	P2 Sg	-	-
Perf.	P2 Pl	P3 Sg Fem	P2 Pl	-	-
Perf.	P2 Pl	P3 Sg Fem	P3 Sg Masc	*ftaħtuhielu*	*ftaħtuhielux*
Perf.	P2 Pl	P3 Sg Fem	P3 Sg Fem	*ftaħtuhielha*	*ftaħtuhilhiex*
Perf.	P2 Pl	P3 Sg Fem	P3 Pl	*ftaħtuhielhom*	*ftaħtuhielhomx*
Perf.	P2 Pl	P3 Pl	P1 Sg	*ftaħtuhomli*	*ftaħtuhomlix*
Perf.	P2 Pl	P3 Pl	P1 Pl	*ftaħtuhomlna*	*ftaħtuhomlniex*
Perf.	P2 Pl	P3 Pl	P2 Sg	-	-
Perf.	P2 Pl	P3 Pl	P2 Pl	-	-
Perf.	P2 Pl	P3 Pl	P3 Sg Masc	*ftaħtuhomlu*	*ftaħtuhomlux*
Perf.	P2 Pl	P3 Pl	P3 Sg Fem	*ftaħtuhomlha*	*ftaħtuhomlhiex*
Perf.	P2 Pl	P3 Pl	P3 Pl	*ftaħtuhomlhom*	*ftaħtuhomlhomx*
Perf.	P3 Sg Masc	-	-	*fetaħ*	*fetaħx*
Perf.	P3 Sg Masc	P1 Sg	-	*fetaħni*	*fetaħnix*
Perf.	P3 Sg Masc	P1 Pl	-	*fetaħna*	*fetaħniex*
Perf.	P3 Sg Masc	P2 Sg	-	*feṫek*	*feṫekx*
Perf.	P3 Sg Masc	P2 Pl	-	*fetaħkom*	*fetaħkomx*
Perf.	P3 Sg Masc	P3 Sg Masc	-	*feṫu*	*feṫux*
Perf.	P3 Sg Masc	P3 Sg Fem	-	*fetaħha*	*fetaħhiex*
Perf.	P3 Sg Masc	P3 Pl	-	*fetaħhom*	*fetaħhomx*
Perf.	P3 Sg Masc	-	P1 Sg	*fetaħli*	*fetaħlix*
Perf.	P3 Sg Masc	-	P1 Pl	*feṫilna*	*feṫilniex*
Perf.	P3 Sg Masc	-	P2 Sg	*fetaħlek*	*fetaħlekx*
Perf.	P3 Sg Masc	-	P2 Pl	*feṫilkom*	*feṫilkomx*
Perf.	P3 Sg Masc	-	P3 Sg Masc	*fetaħlu*	*fetaħlux*
Perf.	P3 Sg Masc	-	P3 Sg Fem	*feṫilha*	*feṫilhiex*
Perf.	P3 Sg Masc	-	P3 Pl	*feṫilhom*	*feṫilhomx*
Perf.	P3 Sg Masc	P3 Sg Masc	P1 Sg	*fetaħhuli*	*fetaħhulix*
Perf.	P3 Sg Masc	P3 Sg Masc	P1 Pl	*fetaħhulna*	*fetaħhulniex*
Perf.	P3 Sg Masc	P3 Sg Masc	P2 Sg	*fetaħhulek*	*fetaħhulekx*
Perf.	P3 Sg Masc	P3 Sg Masc	P2 Pl	*fetaħhulkom*	*fetaħhulkomx*
Perf.	P3 Sg Masc	P3 Sg Masc	P3 Sg Masc	*fetaħhulu*	*fetaħhulux*
Perf.	P3 Sg Masc	P3 Sg Masc	P3 Sg Fem	*fetaħhulha*	*fetaħhulhiex*
Perf.	P3 Sg Masc	P3 Sg Masc	P3 Pl	*fetaħhulhom*	*fetaħhulhomx*

Continued on next page

Perf.	P3 Sg Masc	P3 Sg Fem	P1 Sg	*fetaħhieli*	*fetaħhielix*
Perf.	P3 Sg Masc	P3 Sg Fem	P1 Pl	*fetaħhielna*	*fetaħhilniex*
Perf.	P3 Sg Masc	P3 Sg Fem	P2 Sg	*fetaħhielek*	*fetaħhielekx*
Perf.	P3 Sg Masc	P3 Sg Fem	P2 Pl	*fetaħhielkom*	*fetaħhielkomx*
Perf.	P3 Sg Masc	P3 Sg Fem	P3 Sg Masc	*fetaħhielu*	*fetaħhielux*
Perf.	P3 Sg Masc	P3 Sg Fem	P3 Sg Fem	*fetaħhielha*	*fetaħhilhiex*
Perf.	P3 Sg Masc	P3 Sg Fem	P3 Pl	*fetaħhielhom*	*fetaħhielhomx*
Perf.	P3 Sg Masc	P3 Pl	P1 Sg	*fetaħhomli*	*fetaħhomlix*
Perf.	P3 Sg Masc	P3 Pl	P1 Pl	*fetaħhomlna*	*fetaħhomlniex*
Perf.	P3 Sg Masc	P3 Pl	P2 Sg	*fetaħhomlok*	*fetaħhomlokx*
Perf.	P3 Sg Masc	P3 Pl	P2 Pl	*fetaħhomlkom*	*fetaħhomlkomx*
Perf.	P3 Sg Masc	P3 Pl	P3 Sg Masc	*fetaħhomlu*	*fetaħhomlux*
Perf.	P3 Sg Masc	P3 Pl	P3 Sg Fem	*fetaħhomlha*	*fetaħhomlhiex*
Perf.	P3 Sg Masc	P3 Pl	P3 Pl	*fetaħhomlhom*	*fetaħhomlhomx*
Perf.	P3 Sg Fem	-	-	*fetħet*	*fetħitx*
Perf.	P3 Sg Fem	P1 Sg	-	*fetħitni*	*fetħitnix*
Perf.	P3 Sg Fem	P1 Pl	-	*fetħitna*	*fetħitniex*
Perf.	P3 Sg Fem	P2 Sg	-	*fetħitek*	*fetħitekx*
Perf.	P3 Sg Fem	P2 Pl	-	*fetħitkom*	*fetħitkomx*
Perf.	P3 Sg Fem	P3 Sg Masc	-	*fetħitu*	*fetħitux*
Perf.	P3 Sg Fem	P3 Sg Fem	-	*fetħitha*	*fetħithiex*
Perf.	P3 Sg Fem	P3 Pl	-	*fetħithom*	*fetħithomx*
Perf.	P3 Sg Fem	-	P1 Sg	*fetħitli*	*fetħitlix*
Perf.	P3 Sg Fem	-	P1 Pl	*fetħitilna*	*fetħitilniex*
Perf.	P3 Sg Fem	-	P2 Sg	*fetħitlek*	*fetħitlekx*
Perf.	P3 Sg Fem	-	P2 Pl	*fetħitilkom*	*fetħitilkomx*
Perf.	P3 Sg Fem	-	P3 Sg Masc	*fetħitlu*	*fetħitlux*
Perf.	P3 Sg Fem	-	P3 Sg Fem	*fetħitilha*	*fetħitilhiex*
Perf.	P3 Sg Fem	-	P3 Pl	*fetħitilhom*	*fetħitilhomx*
Perf.	P3 Sg Fem	P3 Sg Masc	P1 Sg	*fetħithuli*	*fetħithulix*
Perf.	P3 Sg Fem	P3 Sg Masc	P1 Pl	*fetħithulna*	*fetħithulniex*
Perf.	P3 Sg Fem	P3 Sg Masc	P2 Sg	*fetħithulek*	*fetħithulekx*
Perf.	P3 Sg Fem	P3 Sg Masc	P2 Pl	*fetħithulkom*	*fetħithulkomx*
Perf.	P3 Sg Fem	P3 Sg Masc	P3 Sg Masc	*fetħithulu*	*fetħithulux*
Perf.	P3 Sg Fem	P3 Sg Masc	P3 Sg Fem	*fetħithulha*	*fetħithulhiex*
Perf.	P3 Sg Fem	P3 Sg Masc	P3 Pl	*fetħithulhom*	*fetħithulhomx*
Perf.	P3 Sg Fem	P3 Sg Fem	P1 Sg	*fetħithieli*	*fetħithielix*
Perf.	P3 Sg Fem	P3 Sg Fem	P1 Pl	*fetħithielna*	*fetħithilniex*

Continued on next page

Perf.	P3 Sg Fem	P3 Sg Fem	P2 Sg	*fɬithielek*	*fɬithielekx*
Perf.	P3 Sg Fem	P3 Sg Fem	P2 Pl	*fɬithielkom*	*fɬithielkomx*
Perf.	P3 Sg Fem	P3 Sg Fem	P3 Sg Masc	*fɬithielu*	*fɬithielux*
Perf.	P3 Sg Fem	P3 Sg Fem	P3 Sg Fem	*fɬithielha*	*fɬithilhiex*
Perf.	P3 Sg Fem	P3 Sg Fem	P3 Pl	*fɬithielhom*	*fɬithielhomx*
Perf.	P3 Sg Fem	P3 Pl	P1 Sg	*fɬithomli*	*fɬithomlix*
Perf.	P3 Sg Fem	P3 Pl	P1 Pl	*fɬithomlna*	*fɬithomlniex*
Perf.	P3 Sg Fem	P3 Pl	P2 Sg	*fɬithomlok*	*fɬithomlokx*
Perf.	P3 Sg Fem	P3 Pl	P2 Pl	*fɬithomlkom*	*fɬithomlkomx*
Perf.	P3 Sg Fem	P3 Pl	P3 Sg Masc	*fɬithomlu*	*fɬithomlux*
Perf.	P3 Sg Fem	P3 Pl	P3 Sg Fem	*fɬithomlha*	*fɬithomlhiex*
Perf.	P3 Sg Fem	P3 Pl	P3 Pl	*fɬithomlhom*	*fɬithomlhomx*
Perf.	P3 Pl	-	-	*fɬhu*	*fɬhux*
Perf.	P3 Pl	P1 Sg	-	*fɬhuni*	*fɬhunix*
Perf.	P3 Pl	P1 Pl	-	*fɬhuna*	*fɬhuniex*
Perf.	P3 Pl	P2 Sg	-	*fɬhuk*	*fɬhukx*
Perf.	P3 Pl	P2 Pl	-	*fɬhukom*	*fɬhukomx*
Perf.	P3 Pl	P3 Sg Masc	-	*fɬhuh*	*fɬhuhx*
Perf.	P3 Pl	P3 Sg Fem	-	*fɬhuha*	*fɬhuhiex*
Perf.	P3 Pl	P3 Pl	-	*fɬhuhom*	*fɬhuhomx*
Perf.	P3 Pl	-	P1 Sg	*fɬhuli*	*fɬhulix*
Perf.	P3 Pl	-	P1 Pl	*fɬhulna*	*fɬhulniex*
Perf.	P3 Pl	-	P2 Sg	*fɬhulek*	*fɬhulekx*
Perf.	P3 Pl	-	P2 Pl	*fɬhulkom*	*fɬhulkomx*
Perf.	P3 Pl	-	P3 Sg Masc	*fɬhulu*	*fɬhulux*
Perf.	P3 Pl	-	P3 Sg Fem	*fɬhulha*	*fɬhulhiex*
Perf.	P3 Pl	-	P3 Pl	*fɬhulhom*	*fɬhulhomx*
Perf.	P3 Pl	P3 Sg Masc	P1 Sg	*fɬhuhuli*	*fɬhuhulix*
Perf.	P3 Pl	P3 Sg Masc	P1 Pl	*fɬhuhulna*	*fɬhuhulniex*
Perf.	P3 Pl	P3 Sg Masc	P2 Sg	*fɬhuhulek*	*fɬhuhulekx*
Perf.	P3 Pl	P3 Sg Masc	P2 Pl	*fɬhuhulkom*	*fɬhuhulkomx*
Perf.	P3 Pl	P3 Sg Masc	P3 Sg Masc	*fɬhuhulu*	*fɬhuhulux*
Perf.	P3 Pl	P3 Sg Masc	P3 Sg Fem	*fɬhuhulha*	*fɬhuhulhiex*
Perf.	P3 Pl	P3 Sg Masc	P3 Pl	*fɬhuhulhom*	*fɬhuhulhomx*
Perf.	P3 Pl	P3 Sg Fem	P1 Sg	*fɬhuhieli*	*fɬhuhielix*
Perf.	P3 Pl	P3 Sg Fem	P1 Pl	*fɬhuhielna*	*fɬhuhilniex*
Perf.	P3 Pl	P3 Sg Fem	P2 Sg	*fɬhuhielek*	*fɬhuhielekx*
Perf.	P3 Pl	P3 Sg Fem	P2 Pl	*fɬhuhielkom*	*fɬhuhielkomx*

Continued on next page

Perf.	P3 Pl	P3 Sg Fem	P3 Sg Masc	*fetɬuhielu*	*fetɬuhielux*
Perf.	P3 Pl	P3 Sg Fem	P3 Sg Fem	*fetɬuhielha*	*fetɬuhilhiex*
Perf.	P3 Pl	P3 Sg Fem	P3 Pl	*fetɬuhielhom*	*fetɬuhielhomx*
Perf.	P3 Pl	P3 Pl	P1 Sg	*fetɬuhomli*	*fetɬuhomlix*
Perf.	P3 Pl	P3 Pl	P1 Pl	*fetɬuhomlna*	*fetɬuhomlniex*
Perf.	P3 Pl	P3 Pl	P2 Sg	*fetɬuhomlok*	*fetɬuhomlokx*
Perf.	P3 Pl	P3 Pl	P2 Pl	*fetɬuhomlkom*	*fetɬuhomlkomx*
Perf.	P3 Pl	P3 Pl	P3 Sg Masc	*fetɬuhomlu*	*fetɬuhomlux*
Perf.	P3 Pl	P3 Pl	P3 Sg Fem	*fetɬuhomlha*	*fetɬuhomlhiex*
Perf.	P3 Pl	P3 Pl	P3 Pl	*fetɬuhomlhom*	*fetɬuhomlhomx*
Impf.	P1 Sg	-	-	*niftaħ*	*niftaħx*
Impf.	P1 Sg	P1 Sg	-	-	-
Impf.	P1 Sg	P1 Pl	-	-	-
Impf.	P1 Sg	P2 Sg	-	*nifɬhek*	*nifɬhekx*
Impf.	P1 Sg	P2 Pl	-	*niftaħkom*	*niftaħkomx*
Impf.	P1 Sg	P3 Sg Masc	-	*nifɬhu*	*nifɬhux*
Impf.	P1 Sg	P3 Sg Fem	-	*niftaħha*	*niftaħhiex*
Impf.	P1 Sg	P3 Pl	-	*niftaħhom*	*niftaħhomx*
Impf.	P1 Sg	-	P1 Sg	-	-
Impf.	P1 Sg	-	P1 Pl	-	-
Impf.	P1 Sg	-	P2 Sg	*niftaħlek*	*niftaħlekx*
Impf.	P1 Sg	-	P2 Pl	*nifɬhilkom*	*nifɬhilkomx*
Impf.	P1 Sg	-	P3 Sg Masc	*niftaħlu*	*niftaħlux*
Impf.	P1 Sg	-	P3 Sg Fem	*nifɬhilha*	*nifɬhilhiex*
Impf.	P1 Sg	-	P3 Pl	*nifɬhilhom*	*nifɬhilhomx*
Impf.	P1 Sg	P3 Sg Masc	P1 Sg	-	-
Impf.	P1 Sg	P3 Sg Masc	P1 Pl	-	-
Impf.	P1 Sg	P3 Sg Masc	P2 Sg	*niftaħhulek*	*niftaħhulekx*
Impf.	P1 Sg	P3 Sg Masc	P2 Pl	*niftaħhulkom*	*niftaħhulkomx*
Impf.	P1 Sg	P3 Sg Masc	P3 Sg Masc	*niftaħhulu*	*niftaħhulux*
Impf.	P1 Sg	P3 Sg Masc	P3 Sg Fem	*niftaħhulha*	*niftaħhulhiex*
Impf.	P1 Sg	P3 Sg Masc	P3 Pl	*niftaħhulhom*	*niftaħhulhomx*
Impf.	P1 Sg	P3 Sg Fem	P1 Sg	-	-
Impf.	P1 Sg	P3 Sg Fem	P1 Pl	-	-
Impf.	P1 Sg	P3 Sg Fem	P2 Sg	*niftaħhielek*	*niftaħhielekx*
Impf.	P1 Sg	P3 Sg Fem	P2 Pl	*niftaħhielkom*	*niftaħhielkomx*
Impf.	P1 Sg	P3 Sg Fem	P3 Sg Masc	*niftaħhielu*	*niftaħhielux*
Impf.	P1 Sg	P3 Sg Fem	P3 Sg Fem	*niftaħhielha*	*niftaħhilhiex*

Continued on next page

Impf.	P1 Sg	P3 Sg Fem	P3 Pl	*niftaħhielhom*	*niftaħhielhomx*
Impf.	P1 Sg	P3 Pl	P1 Sg	-	-
Impf.	P1 Sg	P3 Pl	P1 Pl	-	-
Impf.	P1 Sg	P3 Pl	P2 Sg	*niftaħhomlok*	*niftaħhomlokx*
Impf.	P1 Sg	P3 Pl	P2 Pl	*niftaħhomlkom*	*niftaħhomlkomx*
Impf.	P1 Sg	P3 Pl	P3 Sg Masc	*niftaħhomlu*	*niftaħhomlux*
Impf.	P1 Sg	P3 Pl	P3 Sg Fem	*niftaħhomlha*	*niftaħhomlhiex*
Impf.	P1 Sg	P3 Pl	P3 Pl	*niftaħhomlhom*	*niftaħhomlhomx*
Impf.	P1 Pl	-	-	*nifthu*	*nifthux*
Impf.	P1 Pl	P1 Sg	-	-	-
Impf.	P1 Pl	P1 Pl	-	-	-
Impf.	P1 Pl	P2 Sg	-	*nifthek*	*nifthekx*
Impf.	P1 Pl	P2 Pl	-	*nifthukom*	*nifthukomx*
Impf.	P1 Pl	P3 Sg Masc	-	*nifthuh*	*nifthuhx*
Impf.	P1 Pl	P3 Sg Fem	-	*nifthuha*	*nifthuhiex*
Impf.	P1 Pl	P3 Pl	-	*nifthuhom*	*nifthuhomx*
Impf.	P1 Pl	-	P1 Sg	-	-
Impf.	P1 Pl	-	P1 Pl	-	-
Impf.	P1 Pl	-	P2 Sg	*nifthulek*	*nifthulekx*
Impf.	P1 Pl	-	P2 Pl	*nifthulkom*	*nifthulkomx*
Impf.	P1 Pl	-	P3 Sg Masc	*nifthulu*	*nifthulux*
Impf.	P1 Pl	-	P3 Sg Fem	*nifthulha*	*nifthulhiex*
Impf.	P1 Pl	-	P3 Pl	*nifthulhom*	*nifthulhomx*
Impf.	P1 Pl	P3 Sg Masc	P1 Sg	-	-
Impf.	P1 Pl	P3 Sg Masc	P1 Pl	-	-
Impf.	P1 Pl	P3 Sg Masc	P2 Sg	*nifthuhulek*	*nifthuhulekx*
Impf.	P1 Pl	P3 Sg Masc	P2 Pl	*nifthuhulkom*	*nifthuhulkomx*
Impf.	P1 Pl	P3 Sg Masc	P3 Sg Masc	*nifthuhulu*	*nifthuhulux*
Impf.	P1 Pl	P3 Sg Masc	P3 Sg Fem	*nifthuhulha*	*nifthuhulhiex*
Impf.	P1 Pl	P3 Sg Masc	P3 Pl	*nifthuhulhom*	*nifthuhulhomx*
Impf.	P1 Pl	P3 Sg Fem	P1 Sg	-	-
Impf.	P1 Pl	P3 Sg Fem	P1 Pl	-	-
Impf.	P1 Pl	P3 Sg Fem	P2 Sg	*nifthuhielek*	*nifthuhielekx*
Impf.	P1 Pl	P3 Sg Fem	P2 Pl	*nifthuhielkom*	*nifthuhielkomx*
Impf.	P1 Pl	P3 Sg Fem	P3 Sg Masc	*nifthuhielu*	*nifthuhielux*
Impf.	P1 Pl	P3 Sg Fem	P3 Sg Fem	*nifthuhielha*	*nifthuhilhiex*
Impf.	P1 Pl	P3 Sg Fem	P3 Pl	*nifthuhielhom*	*nifthuhielhomx*
Impf.	P1 Pl	P3 Pl	P1 Sg	-	-

Continued on next page

Impf.	P1 Pl	P3 Pl	P1 Pl	-	-
Impf.	P1 Pl	P3 Pl	P2 Sg	*nifthuhomlok*	*nifthuhomlokx*
Impf.	P1 Pl	P3 Pl	P2 Pl	*nifthuhomlkom*	*nifthuhomlkomx*
Impf.	P1 Pl	P3 Pl	P3 Sg Masc	*nifthuhomlu*	*nifthuhomlux*
Impf.	P1 Pl	P3 Pl	P3 Sg Fem	*nifthuhomlha*	*nifthuhomlhiex*
Impf.	P1 Pl	P3 Pl	P3 Pl	*nifthuhomlhom*	*nifthuhomlhomx*
Impf.	P2 Sg	-	-	*tiftaħ*	*tiftaħx*
Impf.	P2 Sg	P1 Sg	-	*tiftaħni*	*tiftaħnix*
Impf.	P2 Sg	P1 Pl	-	*tiftaħna*	*tiftaħniex*
Impf.	P2 Sg	P2 Sg	-	-	-
Impf.	P2 Sg	P2 Pl	-	-	-
Impf.	P2 Sg	P3 Sg Masc	-	*tifthu*	*tifthux*
Impf.	P2 Sg	P3 Sg Fem	-	*tiftaħha*	*tiftaħhiex*
Impf.	P2 Sg	P3 Pl	-	*tiftaħhom*	*tiftaħhomx*
Impf.	P2 Sg	-	P1 Sg	*tiftaħli*	*tiftaħlix*
Impf.	P2 Sg	-	P1 Pl	*tifthilna*	*tifthilniex*
Impf.	P2 Sg	-	P2 Sg	-	-
Impf.	P2 Sg	-	P2 Pl	-	-
Impf.	P2 Sg	-	P3 Sg Masc	*tiftaħlu*	*tiftaħlux*
Impf.	P2 Sg	-	P3 Sg Fem	*tifthilha*	*tifthilhiex*
Impf.	P2 Sg	-	P3 Pl	*tifthilhom*	*tifthilhomx*
Impf.	P2 Sg	P3 Sg Masc	P1 Sg	*tiftaħhuli*	*tiftaħhulix*
Impf.	P2 Sg	P3 Sg Masc	P1 Pl	*tiftaħhulna*	*tiftaħhulniex*
Impf.	P2 Sg	P3 Sg Masc	P2 Sg	-	-
Impf.	P2 Sg	P3 Sg Masc	P2 Pl	-	-
Impf.	P2 Sg	P3 Sg Masc	P3 Sg Masc	*tiftaħhulu*	*tiftaħhulux*
Impf.	P2 Sg	P3 Sg Masc	P3 Sg Fem	*tiftaħhulha*	*tiftaħhulhiex*
Impf.	P2 Sg	P3 Sg Masc	P3 Pl	*tiftaħhulhom*	*tiftaħhulhomx*
Impf.	P2 Sg	P3 Sg Fem	P1 Sg	*tiftaħhieli*	*tiftaħhielix*
Impf.	P2 Sg	P3 Sg Fem	P1 Pl	*tiftaħhielna*	*tiftaħhilniex*
Impf.	P2 Sg	P3 Sg Fem	P2 Sg	-	-
Impf.	P2 Sg	P3 Sg Fem	P2 Pl	-	-
Impf.	P2 Sg	P3 Sg Fem	P3 Sg Masc	*tiftaħhielu*	*tiftaħhielux*
Impf.	P2 Sg	P3 Sg Fem	P3 Sg Fem	*tiftaħhielha*	*tiftaħhilhiex*
Impf.	P2 Sg	P3 Sg Fem	P3 Pl	*tiftaħhielhom*	*tiftaħhielhomx*
Impf.	P2 Sg	P3 Pl	P1 Sg	*tiftaħhomli*	*tiftaħhomlix*
Impf.	P2 Sg	P3 Pl	P1 Pl	*tiftaħhomlna*	*tiftaħhomlniex*
Impf.	P2 Sg	P3 Pl	P2 Sg	-	-

Continued on next page

Impf.	P2 Sg	P3 Pl	P2 Pl	-	-
Impf.	P2 Sg	P3 Pl	P3 Sg Masc	*tiftaħhomlu*	*tiftaħhomlux*
Impf.	P2 Sg	P3 Pl	P3 Sg Fem	*tiftaħhomlha*	*tiftaħhomlhiex*
Impf.	P2 Sg	P3 Pl	P3 Pl	*tiftaħhomlhom*	*tiftaħhomlhomx*
Impf.	P2 Pl	-	-	*tifthu*	*tifthux*
Impf.	P2 Pl	P1 Sg	-	*tifthuni*	*tifthunix*
Impf.	P2 Pl	P1 Pl	-	*tifthuna*	*tifthuniex*
Impf.	P2 Pl	P2 Sg	-	-	-
Impf.	P2 Pl	P2 Pl	-	-	-
Impf.	P2 Pl	P3 Sg Masc	-	*tifthuh*	*tifthuhx*
Impf.	P2 Pl	P3 Sg Fem	-	*tifthuha*	*tifthuhiex*
Impf.	P2 Pl	P3 Pl	-	*tifthuhom*	*tifthuhomx*
Impf.	P2 Pl	-	P1 Sg	*tifthuli*	*tifthulix*
Impf.	P2 Pl	-	P1 Pl	*tifthula*	*tifthuliex*
Impf.	P2 Pl	-	P2 Sg	-	-
Impf.	P2 Pl	-	P2 Pl	-	-
Impf.	P2 Pl	-	P3 Sg Masc	*tifthulu*	*tifthulux*
Impf.	P2 Pl	-	P3 Sg Fem	*tifthulha*	*tifthulhiex*
Impf.	P2 Pl	-	P3 Pl	*tifthulhom*	*tifthulhomx*
Impf.	P2 Pl	P3 Sg Masc	P1 Sg	*tifthuhuli*	*tifthuhulix*
Impf.	P2 Pl	P3 Sg Masc	P1 Pl	*tifthuhulna*	*tifthuhulniex*
Impf.	P2 Pl	P3 Sg Masc	P2 Sg	-	-
Impf.	P2 Pl	P3 Sg Masc	P2 Pl	-	-
Impf.	P2 Pl	P3 Sg Masc	P3 Sg Masc	*tifthuhulu*	*tifthuhulux*
Impf.	P2 Pl	P3 Sg Masc	P3 Sg Fem	*tifthuhulha*	*tifthuhulhiex*
Impf.	P2 Pl	P3 Sg Masc	P3 Pl	*tifthuhulhom*	*tifthuhulhomx*
Impf.	P2 Pl	P3 Sg Fem	P1 Sg	*tifthuhieli*	*tifthuhielix*
Impf.	P2 Pl	P3 Sg Fem	P1 Pl	*tifthuhielna*	*tifthuhilniex*
Impf.	P2 Pl	P3 Sg Fem	P2 Sg	-	-
Impf.	P2 Pl	P3 Sg Fem	P2 Pl	-	-
Impf.	P2 Pl	P3 Sg Fem	P3 Sg Masc	*tifthuhielu*	*tifthuhielux*
Impf.	P2 Pl	P3 Sg Fem	P3 Sg Fem	*tifthuhielha*	*tifthuhilhiex*
Impf.	P2 Pl	P3 Sg Fem	P3 Pl	*tifthuhielhom*	*tifthuhielhomx*
Impf.	P2 Pl	P3 Pl	P1 Sg	*tifthuhomli*	*tifthuhomlix*
Impf.	P2 Pl	P3 Pl	P1 Pl	*tifthuhomlna*	*tifthuhomlniex*
Impf.	P2 Pl	P3 Pl	P2 Sg	-	-
Impf.	P2 Pl	P3 Pl	P2 Pl	-	-
Impf.	P2 Pl	P3 Pl	P3 Sg Masc	*tifthuhomlu*	*tifthuhomlux*

Continued on next page

Impf.	P2 Pl	P3 Pl	P3 Sg Fem	*tifthuhomlha*	*tifthuhomlhiex*
Impf.	P2 Pl	P3 Pl	P3 Pl	*tifthuhomlhom*	*tifthuhomlhomx*
Impf.	P3 Sg Masc	-	-	*jiftaħ*	*jiftaħx*
Impf.	P3 Sg Masc	P1 Sg	-	*jiftaħni*	*jiftaħnix*
Impf.	P3 Sg Masc	P1 Pl	-	*jiftaħna*	*jiftaħniex*
Impf.	P3 Sg Masc	P2 Sg	-	*jifthek*	*jifthekx*
Impf.	P3 Sg Masc	P2 Pl	-	*jiftaħkom*	*jiftaħkomx*
Impf.	P3 Sg Masc	P3 Sg Masc	-	*jifthu*	*jifthux*
Impf.	P3 Sg Masc	P3 Sg Fem	-	*jiftaħha*	*jiftaħhiex*
Impf.	P3 Sg Masc	P3 Pl	-	*jiftaħhom*	*jiftaħhomx*
Impf.	P3 Sg Masc	-	P1 Sg	*jiftaħli*	*jiftaħlix*
Impf.	P3 Sg Masc	-	P1 Pl	*jifthilna*	*jifthilniex*
Impf.	P3 Sg Masc	-	P2 Sg	*jiftaħlek*	*jiftaħlekx*
Impf.	P3 Sg Masc	-	P2 Pl	*jifthilkom*	*jifthilkomx*
Impf.	P3 Sg Masc	-	P3 Sg Masc	*jiftaħlu*	*jiftaħlux*
Impf.	P3 Sg Masc	-	P3 Sg Fem	*jifthilha*	*jifthilhiex*
Impf.	P3 Sg Masc	-	P3 Pl	*jifthilhom*	*jifthilhomx*
Impf.	P3 Sg Masc	P3 Sg Masc	P1 Sg	*jiftaħhuli*	*jiftaħhulix*
Impf.	P3 Sg Masc	P3 Sg Masc	P1 Pl	*jiftaħhulna*	*jiftaħhulniex*
Impf.	P3 Sg Masc	P3 Sg Masc	P2 Sg	*jiftaħhulek*	*jiftaħhulekx*
Impf.	P3 Sg Masc	P3 Sg Masc	P2 Pl	*jiftaħhulkom*	*jiftaħhulkomx*
Impf.	P3 Sg Masc	P3 Sg Masc	P3 Sg Masc	*jiftaħhulu*	*jiftaħhulux*
Impf.	P3 Sg Masc	P3 Sg Masc	P3 Sg Fem	*jiftaħhulha*	*jiftaħhulhiex*
Impf.	P3 Sg Masc	P3 Sg Masc	P3 Pl	*jiftaħhulhom*	*jiftaħhulhomx*
Impf.	P3 Sg Masc	P3 Sg Fem	P1 Sg	*jiftaħhieli*	*jiftaħhielix*
Impf.	P3 Sg Masc	P3 Sg Fem	P1 Pl	*jiftaħhielna*	*jiftaħhilniex*
Impf.	P3 Sg Masc	P3 Sg Fem	P2 Sg	*jiftaħhielek*	*jiftaħhielekx*
Impf.	P3 Sg Masc	P3 Sg Fem	P2 Pl	*jiftaħhielkom*	*jiftaħhielkomx*
Impf.	P3 Sg Masc	P3 Sg Fem	P3 Sg Masc	*jiftaħhielu*	*jiftaħhielux*
Impf.	P3 Sg Masc	P3 Sg Fem	P3 Sg Fem	*jiftaħhielha*	*jiftaħhilhiex*
Impf.	P3 Sg Masc	P3 Sg Fem	P3 Pl	*jiftaħhielhom*	*jiftaħhielhomx*
Impf.	P3 Sg Masc	P3 Pl	P1 Sg	*jiftaħhomli*	*jiftaħhomlix*
Impf.	P3 Sg Masc	P3 Pl	P1 Pl	*jiftaħhomlna*	*jiftaħhomlniex*
Impf.	P3 Sg Masc	P3 Pl	P2 Sg	*jiftaħhomlok*	*jiftaħhomlokx*
Impf.	P3 Sg Masc	P3 Pl	P2 Pl	*jiftaħhomlkom*	*jiftaħhomlkomx*
Impf.	P3 Sg Masc	P3 Pl	P3 Sg Masc	*jiftaħhomlu*	*jiftaħhomlux*
Impf.	P3 Sg Masc	P3 Pl	P3 Sg Fem	*jiftaħhomlha*	*jiftaħhomlhiex*
Impf.	P3 Sg Masc	P3 Pl	P3 Pl	*jiftaħhomlhom*	*jiftaħhomlhomx*

Continued on next page

Impf.	P3 Sg Fem	-	-	*tiftaħ*	*tiftaħx*
Impf.	P3 Sg Fem	P1 Sg	-	*tiftaħni*	*tiftaħnix*
Impf.	P3 Sg Fem	P1 Pl	-	*tiftaħna*	*tiftaħniex*
Impf.	P3 Sg Fem	P2 Sg	-	*iftħek*	*iftħekx*
Impf.	P3 Sg Fem	P2 Pl	-	*tiftaħkom*	*tiftaħkomx*
Impf.	P3 Sg Fem	P3 Sg Masc	-	*iftħu*	*iftħux*
Impf.	P3 Sg Fem	P3 Sg Fem	-	*tiftaħha*	*tiftaħhiex*
Impf.	P3 Sg Fem	P3 Pl	-	*tiftaħhom*	*tiftaħhomx*
Impf.	P3 Sg Fem	-	P1 Sg	*tiftaħli*	*tiftaħlix*
Impf.	P3 Sg Fem	-	P1 Pl	*iftħilna*	*iftħilniex*
Impf.	P3 Sg Fem	-	P2 Sg	*tiftaħlek*	*tiftaħlekx*
Impf.	P3 Sg Fem	-	P2 Pl	*iftħilkom*	*iftħilkomx*
Impf.	P3 Sg Fem	-	P3 Sg Masc	*tiftaħlu*	*tiftaħlux*
Impf.	P3 Sg Fem	-	P3 Sg Fem	*iftħilha*	*iftħilhiex*
Impf.	P3 Sg Fem	-	P3 Pl	*tiftaħilhom*	*tiftaħilhomx*
Impf.	P3 Sg Fem	P3 Sg Masc	P1 Sg	*tiftaħhuli*	*tiftaħhulix*
Impf.	P3 Sg Fem	P3 Sg Masc	P1 Pl	*tiftaħhulna*	*tiftaħhulniex*
Impf.	P3 Sg Fem	P3 Sg Masc	P2 Sg	*tiftaħhulek*	*tiftaħhulekx*
Impf.	P3 Sg Fem	P3 Sg Masc	P2 Pl	*tiftaħhulkom*	*tiftaħhulkomx*
Impf.	P3 Sg Fem	P3 Sg Masc	P3 Sg Masc	*tiftaħhulu*	*tiftaħhulux*
Impf.	P3 Sg Fem	P3 Sg Masc	P3 Sg Fem	*tiftaħhulha*	*tiftaħhulhiex*
Impf.	P3 Sg Fem	P3 Sg Masc	P3 Pl	*tiftaħhulhom*	*tiftaħhulhomx*
Impf.	P3 Sg Fem	P3 Sg Fem	P1 Sg	*tiftaħhieli*	*tiftaħhielix*
Impf.	P3 Sg Fem	P3 Sg Fem	P1 Pl	*tiftaħhielna*	*tiftaħhilniex*
Impf.	P3 Sg Fem	P3 Sg Fem	P2 Sg	*tiftaħhielek*	*tiftaħhielekx*
Impf.	P3 Sg Fem	P3 Sg Fem	P2 Pl	*tiftaħhielkom*	*tiftaħhielkomx*
Impf.	P3 Sg Fem	P3 Sg Fem	P3 Sg Masc	*tiftaħhielu*	*tiftaħhielux*
Impf.	P3 Sg Fem	P3 Sg Fem	P3 Sg Fem	*tiftaħhielha*	*tiftaħhilhiex*
Impf.	P3 Sg Fem	P3 Sg Fem	P3 Pl	*tiftaħhielhom*	*tiftaħhielhomx*
Impf.	P3 Sg Fem	P3 Pl	P1 Sg	*tiftaħhomli*	*tiftaħhomlix*
Impf.	P3 Sg Fem	P3 Pl	P1 Pl	*tiftaħhomlna*	*tiftaħhomlniex*
Impf.	P3 Sg Fem	P3 Pl	P2 Sg	*tiftaħhomlok*	*tiftaħhomlokx*
Impf.	P3 Sg Fem	P3 Pl	P2 Pl	*tiftaħhomlkom*	*tiftaħhomlkomx*
Impf.	P3 Sg Fem	P3 Pl	P3 Sg Masc	*tiftaħhomlu*	*tiftaħhomlux*
Impf.	P3 Sg Fem	P3 Pl	P3 Sg Fem	*tiftaħhomlha*	*tiftaħhomlhiex*
Impf.	P3 Sg Fem	P3 Pl	P3 Pl	*tiftaħhomlhom*	*tiftaħhomlhomx*
Impf.	P3 Pl	-	-	*jiftħu*	*jiftħux*
Impf.	P3 Pl	P1 Sg	-	*jiftħuni*	*jiftħunix*

Continued on next page

Impf.	P3 Pl	P1 Pl	-	*jifthuna*	*jifthuniex*
Impf.	P3 Pl	P2 Sg	-	*jifthuk*	*jifthukx*
Impf.	P3 Pl	P2 Pl	-	*jifthukom*	*jifthukomx*
Impf.	P3 Pl	P3 Sg Masc	-	*jifthuh*	*jifthuhx*
Impf.	P3 Pl	P3 Sg Fem	-	*jifthuha*	*jifthuhiex*
Impf.	P3 Pl	P3 Pl	-	*jifthuhom*	*jifthuhomx*
Impf.	P3 Pl	-	P1 Sg	*jifthuli*	*jifthulix*
Impf.	P3 Pl	-	P1 Pl	*jifthulna*	*jifthulniex*
Impf.	P3 Pl	-	P2 Sg	*jifthulek*	*jifthulekx*
Impf.	P3 Pl	-	P2 Pl	*jifthulkom*	*jifthulkomx*
Impf.	P3 Pl	-	P3 Sg Masc	*jifthulu*	*jifthulux*
Impf.	P3 Pl	-	P3 Sg Fem	*jifthulha*	*jifthulhiex*
Impf.	P3 Pl	-	P3 Pl	*jifthulhom*	*jifthulhomx*
Impf.	P3 Pl	P3 Sg Masc	P1 Sg	*jifthuhuli*	*jifthuhulix*
Impf.	P3 Pl	P3 Sg Masc	P1 Pl	*jifthuhulna*	*jifthuhulniex*
Impf.	P3 Pl	P3 Sg Masc	P2 Sg	*jifthuhulek*	*jifthuhulekx*
Impf.	P3 Pl	P3 Sg Masc	P2 Pl	*jifthuhulkom*	*jifthuhulkomx*
Impf.	P3 Pl	P3 Sg Masc	P3 Sg Masc	*jifthuhulu*	*jifthuhulux*
Impf.	P3 Pl	P3 Sg Masc	P3 Sg Fem	*jifthuhulha*	*jifthuhulhiex*
Impf.	P3 Pl	P3 Sg Masc	P3 Pl	*jifthuhulhom*	*jifthuhulhomx*
Impf.	P3 Pl	P3 Sg Fem	P1 Sg	*jifthuhieli*	*jifthuhielix*
Impf.	P3 Pl	P3 Sg Fem	P1 Pl	*jifthuhielna*	*jifthuhilniex*
Impf.	P3 Pl	P3 Sg Fem	P2 Sg	*jifthuhielek*	*jifthuhielekx*
Impf.	P3 Pl	P3 Sg Fem	P2 Pl	*jifthuhielkom*	*jifthuhielkomx*
Impf.	P3 Pl	P3 Sg Fem	P3 Sg Masc	*jifthuhielu*	*jifthuhielux*
Impf.	P3 Pl	P3 Sg Fem	P3 Sg Fem	*jifthuhielha*	*jifthuhilhiex*
Impf.	P3 Pl	P3 Sg Fem	P3 Pl	*jifthuhielhom*	*jifthuhielhomx*
Impf.	P3 Pl	P3 Pl	P1 Sg	*jifthuhomli*	*jifthuhomlix*
Impf.	P3 Pl	P3 Pl	P1 Pl	*jifthuhomlna*	*jifthuhomlniex*
Impf.	P3 Pl	P3 Pl	P2 Sg	*jifthuhomlok*	*jifthuhomlokx*
Impf.	P3 Pl	P3 Pl	P2 Pl	*jifthuhomlkom*	*jifthuhomlkomx*
Impf.	P3 Pl	P3 Pl	P3 Sg Masc	*jifthuhomlu*	*jifthuhomlux*
Impf.	P3 Pl	P3 Pl	P3 Sg Fem	*jifthuhomlha*	*jifthuhomlhiex*
Impf.	P3 Pl	P3 Pl	P3 Pl	*jifthuhomlhom*	*jifthuhomlhomx*
Imp.	P2 Sg	-	-	*iftaħ*	*iftaħx*
Imp.	P2 Sg	P1 Sg	-	*iftaħni*	*iftaħnix*
Imp.	P2 Sg	P1 Pl	-	*iftaħna*	*iftaħniex*
Imp.	P2 Sg	P2 Sg	-	-	-

Continued on next page

Imp.	P2 Sg	P2 Pl	-	-	-
Imp.	P2 Sg	P3 Sg Masc	-	*ifthu*	*ifthux*
Imp.	P2 Sg	P3 Sg Fem	-	*iftaħha*	*iftaħhiex*
Imp.	P2 Sg	P3 Pl	-	*iftaħhom*	*iftaħhomx*
Imp.	P2 Sg	-	P1 Sg	*iftaħli*	*iftaħlix*
Imp.	P2 Sg	-	P1 Pl	*ifthilna*	*ifthilniex*
Imp.	P2 Sg	-	P2 Sg	-	-
Imp.	P2 Sg	-	P2 Pl	-	-
Imp.	P2 Sg	-	P3 Sg Masc	*iftaħlu*	*iftaħlux*
Imp.	P2 Sg	-	P3 Sg Fem	*ifthilha*	*ifthilhiex*
Imp.	P2 Sg	-	P3 Pl	*ifthilhom*	*ifthilhomx*
Imp.	P2 Sg	P3 Sg Masc	P1 Sg	*iftaħhuli*	*iftaħhulix*
Imp.	P2 Sg	P3 Sg Masc	P1 Pl	*iftaħhulna*	*iftaħhulniex*
Imp.	P2 Sg	P3 Sg Masc	P2 Sg	-	-
Imp.	P2 Sg	P3 Sg Masc	P2 Pl	-	-
Imp.	P2 Sg	P3 Sg Masc	P3 Sg Masc	*iftaħhulu*	*iftaħhulux*
Imp.	P2 Sg	P3 Sg Masc	P3 Sg Fem	*iftaħhulha*	*iftaħhulhiex*
Imp.	P2 Sg	P3 Sg Masc	P3 Pl	*iftaħhulhom*	*iftaħhulhomx*
Imp.	P2 Sg	P3 Sg Fem	P1 Sg	*iftaħhieli*	*iftaħhielix*
Imp.	P2 Sg	P3 Sg Fem	P1 Pl	*iftaħhielna*	*iftaħhilniex*
Imp.	P2 Sg	P3 Sg Fem	P2 Sg	-	-
Imp.	P2 Sg	P3 Sg Fem	P2 Pl	-	-
Imp.	P2 Sg	P3 Sg Fem	P3 Sg Masc	*iftaħhielu*	*iftaħhielux*
Imp.	P2 Sg	P3 Sg Fem	P3 Sg Fem	*iftaħhielha*	*iftaħhilhiex*
Imp.	P2 Sg	P3 Sg Fem	P3 Pl	*iftaħhielhom*	*iftaħhielhomx*
Imp.	P2 Sg	P3 Pl	P1 Sg	*iftaħhomli*	*iftaħhomlix*
Imp.	P2 Sg	P3 Pl	P1 Pl	*iftaħhomlna*	*iftaħhomlniex*
Imp.	P2 Sg	P3 Pl	P2 Sg	-	-
Imp.	P2 Sg	P3 Pl	P2 Pl	-	-
Imp.	P2 Sg	P3 Pl	P3 Sg Masc	*iftaħhomlu*	*iftaħhomlux*
Imp.	P2 Sg	P3 Pl	P3 Sg Fem	*iftaħhomlha*	*iftaħhomlhiex*
Imp.	P2 Sg	P3 Pl	P3 Pl	*iftaħhomlhom*	*iftaħhomlhomx*
Imp.	P2 Pl	-	-	*ifthu*	*ifthux*
Imp.	P2 Pl	P1 Sg	-	*ifthuni*	*ifthunix*
Imp.	P2 Pl	P1 Pl	-	*ifthuna*	*ifthuniex*
Imp.	P2 Pl	P2 Sg	-	-	-
Imp.	P2 Pl	P2 Pl	-	-	-
Imp.	P2 Pl	P3 Sg Masc	-	*ifthuh*	*ifthuhx*

Continued on next page

Imp.	P2 Pl	P3 Sg Fem	-	*ifthuha*	*ifthuhiex*
Imp.	P2 Pl	P3 Pl	-	*ifthuhom*	*ifthuhomx*
Imp.	P2 Pl	-	P1 Sg	*ifthuli*	*ifthulix*
Imp.	P2 Pl	-	P1 Pl	*ifthulna*	*ifthulniex*
Imp.	P2 Pl	-	P2 Sg	-	-
Imp.	P2 Pl	-	P2 Pl	-	-
Imp.	P2 Pl	-	P3 Sg Masc	*ifthulu*	*ifthulux*
Imp.	P2 Pl	-	P3 Sg Fem	*ifthulha*	*ifthulhiex*
Imp.	P2 Pl	-	P3 Pl	*ifthulhom*	*ifthulhomx*
Imp.	P2 Pl	P3 Sg Masc	P1 Sg	*ifthuhuli*	*ifthuhulix*
Imp.	P2 Pl	P3 Sg Masc	P1 Pl	*ifthuhulna*	*ifthuhulniex*
Imp.	P2 Pl	P3 Sg Masc	P2 Sg	-	-
Imp.	P2 Pl	P3 Sg Masc	P2 Pl	-	-
Imp.	P2 Pl	P3 Sg Masc	P3 Sg Masc	*ifthuhulu*	*ifthuhulux*
Imp.	P2 Pl	P3 Sg Masc	P3 Sg Fem	*ifthuhulha*	*ifthuhulhiex*
Imp.	P2 Pl	P3 Sg Masc	P3 Pl	*ifthuhulhom*	*ifthuhulhomx*
Imp.	P2 Pl	P3 Sg Fem	P1 Sg	*ifthuhieli*	*ifthuhielix*
Imp.	P2 Pl	P3 Sg Fem	P1 Pl	*ifthuhielna*	*ifthuhilniex*
Imp.	P2 Pl	P3 Sg Fem	P2 Sg	-	-
Imp.	P2 Pl	P3 Sg Fem	P2 Pl	-	-
Imp.	P2 Pl	P3 Sg Fem	P3 Sg Masc	*ifthuhielu*	*ifthuhielux*
Imp.	P2 Pl	P3 Sg Fem	P3 Sg Fem	*ifthuhielha*	*ifthuhilhiex*
Imp.	P2 Pl	P3 Sg Fem	P3 Pl	*ifthuhielhom*	*ifthuhielhomx*
Imp.	P2 Pl	P3 Pl	P1 Sg	*ifthuhomli*	*ifthuhomlix*
Imp.	P2 Pl	P3 Pl	P1 Pl	*ifthuhomlna*	*ifthuhomlniex*
Imp.	P2 Pl	P3 Pl	P2 Sg	-	-
Imp.	P2 Pl	P3 Pl	P2 Pl	-	-
Imp.	P2 Pl	P3 Pl	P3 Sg Masc	*ifthuhomlu*	*ifthuhomlux*
Imp.	P2 Pl	P3 Pl	P3 Sg Fem	*ifthuhomlha*	*ifthuhomlhiex*
Imp.	P2 Pl	P3 Pl	P3 Pl	*ifthuhomlhom*	*ifthuhomlhomx*

Appendix D

Links, source code and licenses

- The source code for the Maltese resource grammar is obtainable from the main GF repository at: http://www.grammaticalframework.org/lib/src/maltese/. The code is released under the GNU Lesser General Public License[1]. This license allows free use for any purpose, including use within non-open source applications.
- The *Ġabra* web application is accessible at: http://mlrs.research.um.edu.mt/resources/gabra/. Its source code is not made public for security reasons, but it is based on the open-source CakePHP framework and MongoDB database engine.
- The lexicon data collected in *Ġabra* is covered by a CC-BY license[2]. It is accessible via the web application itself, the web service, and downloadable in BSON format.
- The manually-checked gold-standard Maltese verb inflection tables are made available under the MIT license[3] and are available at https://github.com/johnjcamilleri/maltese-verb-inflections/

[1] http://www.gnu.org/licenses/lgpl-3.0.txt, accessed 2013-09-05
[2] http://creativecommons.org/licenses/by/3.0/, accessed 2013-09-05
[3] http://opensource.org/licenses/MIT, accessed 2013-09-05

Glossary

Linguistic abbreviations

anter. anterior

coll. collective

C consonant

C1, C2, C3, C4 first, second, third, fourth radical in a root

DO direct object

f., fem. feminine

I, II, III, IV, V, VI, VII, VIII, IX, X Binyanim forms 1–10

Imp. Imperative mood

IO indirect object

Impf. Imperfective aspect

m., masc. masculine

neg. negative

NP noun phrase

part. participle

Perf. Perfective aspect

pl. plural

S subject

simult. simultaneous (tense)

sg. singular

s.o. someone

s.t. something

O object

P1, P2, P3 first, second, third person

V verb

v vowel

VP verb Phrase

v1, v2 first, second vowel in a pattern

Technical acronyms

API Application Programming Interface

AST Abstract Syntax Tree

BSON Binary JSON

CCG Combinatory Categorial Grammar (Steedman & Baldridge, 2005)

CC-BY Creative Commons Attribution license

CNL Controlled Natural Language

CSV Comma-separated values

DB Database

FM Functional Morphology (Forsberg & Ranta, 2004)

GF Grammatical Framework (Ranta, 2011)

HPSG Head-driven Phrase Structure Grammar (Pollard & Sag, 1994)

JSON JavaScript Object Notation

LFG Lexical Functional Grammar (Kaplan & Bresnan, 1995)

LGPL GNU Lesser General Public License

LST Lexicon Structuring Technique (Dalli, 2002a)

MT Machine Translation

NLP Natural Language Processing

ODL Object Description Language (Rosner *et al.* , 2006)

PMCFG Parallel multiple context-free grammar (Angelov, 2009)

PGF Portable Grammar Format (Angelov, 2011)

RDBMS Relational database management system

regex regular expression

RGL GF Resource Grammar Library (Ranta, 2009)

TAG Tree-Adjoining Grammar (Joshi & Schabes, 1997)

XML Extensible Markup Language

Bibliography

Angelov, Krasimir. 2009. Incremental parsing with parallel multiple context-free grammars. *Pages 69–76 of: Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics (EACL '09)*. Morristown, NJ, USA: Association for Computational Linguistics.

Angelov, Krasimir. 2011. *The Mechanics of the Grammatical Framework*. Ph.D. Thesis, Chalmers University of Technology and Goteborg University.

Angelov, Krasimir, Carlson, Lauri, Enache, Ramona, Listenmaa, Inari, Ranta, Aarne, & Virk, Shafqat Mumtaz. 2013. *Lexicon extraction in MOLTO*.

Aquilina, Joseph. 1987. *Maltese-English Dictionary Vol. I, A-L*. Valletta, Malta: Midsea Books.

Aquilina, Joseph. 1990. *Maltese-English Dictionary Vol. II, M-Z*. Valletta, Malta: Midsea Books.

Attard, Duncan Paul. 2005. *A Lexicon Server Toolkit for Maltese*. Final Year Project, University of Malta.

Azzopardi, Carmel. 2007. *Gwida għall-ortografija*. 2nd edn. Santa Venera, Malta: Klabb Kotba Maltin.

Bender, Emily M, Flickinger, Dan, & Oepen, Stephan. 2002. The grammar matrix: An open-source starter-kit for the rapid development of cross-linguistically consistent broad-coverage precision grammars. *Pages 8–14 of:* Carroll, John, Oostdijk, Nelleke, & Sutcliffe, Richard (eds), *Proceedings of the Workshop on Grammar Engineering and Evaluation at the 19th International Conference on Computational Linguistics*.

Borg, Albert, & Azzopardi-Alexander, Marie. 1997. *Maltese*. London: Routledge.

Borg, Alexander. 1974. Maltese Numerals. *Zeitschrift der Deutschen Morgenländischen Gesellschaft*, **124**, 291.

Brother Henry F.S.C. 1980. *Grammatika Maltija*. 6th edn. Gżira, Malta: De La Salle Brothers.

Camilleri, John J., & Spagnol, Michael. 2013. An online database of root-and-pattern verbs. *In: 4th International Conference on Maltese Linguistics (Lingwistika 2013)*.

Dalli, Angelo. 2002a. *Computational Lexicon for Maltese*. M.Sc. Thesis, University of Malta.

Dalli, Angelo. 2002b. Creation and evaluation of extensible language resources for maltese. *Pages 813–818 of: Proceedings of the Third International Conference on Language Resources and Evaluation (LREC 2002)*.

Dalli, Angelo, Tablan, Valentin, Bontcheva, Kalina, Wilks, Yorick, Broeder, Dan, & Wittenburg, Peter. 2004. Web Services Architecture for Language Resources. *Pages 365–368 of: LREC 2004*.

Détrez, Grégoire, & Ranta, Aarne. 2012. Smart paradigms and the predictability and complexity of inflectional morphology. *In: Proceedings of the 13th Conference of the EACL*. Avignon, France: Association for Computational Linguistics.

Ebert, Karen. 2000. Aspect in Maltese. *Pages 701–733 of:* Dahl, Östen (ed), *Tense and Aspect in the Languages of Europe*, 1 edn. Berlin: Mouton de Gruyter.

Ellul, Leanne. 2013. *Verbal nouns in Maltese*. M.A. Thesis (forthcoming), University of Malta.

European Commission. 2011. *User language preferences online: Analytical report*.

Fabri, Ray. 1995. The tense and aspect system of Maltese. *Pages 327–343 of:* Thieroff, Rolf (ed), *Tense Systems in European Languages II*. De Gruyter.

Fabri, Ray. 2009. Stem allomorphy in the Maltese verb. *Ilsienna*, **1**, 1–20.

Falzon, Grazio. 2012. *Basic English-Maltese Dictionary*.

Forsberg, Markus, & Ranta, Aarne. 2004. Functional morphology. *Page 213 of: Proceedings of the ninth ACM SIGPLAN international conference on Functional programming - ICFP '04*. New York, New York, USA: ACM Press.

Galea, David. 1996. *Morpological Analysis of Maltese Verbs*. Final Year Project, University of Malta.

Gatt, Albert, & Borg, Claudia. 2011. *Using the MLRS Corpus*. Tech. rept. Institute of Linguistics, Univeristy of Malta.

Gatt, Albert, Vella, Alexandra, & Caruana, Joe. 2003. *Annotating textual and speech data in Maltese*. Tech. rept. International Standards Organisation Language Resource Management.

Il-Kunsill Nazzjonali ta' l-Ilsien Malti. 2008. *Innaqqsu l-inċertezzi 2 — Seminar fuq il-kliem ta' nisel Ingliż fil-Malti*.

Joshi, Aravind K., & Schabes, Yves. 1997. Tree-Adjoining Grammars. *Pages 69–123 of:* Rozenberg, Grzegorz, & Salomaa, Arto (eds), *Handbook of Formal Languages*. Springer Berlin Heidelberg.

Kaplan, RM, & Bresnan, Joan. 1995. Lexical-Functional Grammar: A Formal System for Grammatical Representation. *In:* Mary Dalrymple, Zaenen, Annie, III, John Maxwell, & Kaplan, Ronald M. (eds), *Formal Issues in Lexical-Functional Grammar*. CSLI Publications.

Mangion, Gordon K. 1999. *Spelling Correction for Maltese*. Final Year Project, University of Malta.

Mayer, Thomas, Spagnol, Michael, & Schönhuber, Florian. 2013. *Fixing the broken plural in Maltese*.

Mifsud, Manwel. 1995. *Loan verbs in Maltese: A Descriptive and Comparative Study*. Leiden, The Netherlands: E.J. Brill.

MLRS. 2013. *Maltese word list*.

Paggio, Patrizia, Galea, Luke, & Vella, Alexandra. 2013. MAMCO: a Maltese Multimodal Corpus. *In: 4th International Conference on Maltese Linguistics (Lingwistika 2013)*.

Pollard, Carl, & Sag, Ivan A. 1994. *Head-driven phrase structure grammar*. Chicago: University of Chicago Press.

Ralf, Steinberger, Pouliquen, Bruno, Widiger, Anna, Ignat, Camelia, Erjavec, Tomaž, Tufis, Dan, & Varga, Dániel. 2006. The JRC-Acquis: A multilingual aligned parallel corpus with 20+ languages. *In: Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC'2006)*.

Ranta, Aarne. 2009. The GF Resource Grammar Library. *Linguistic Issues in Language Technology*, **2**(2).

Ranta, Aarne. 2011. *Grammatical Framework: Programming with Multilingual Grammars*. Stanford, CA, USA: CSLI Publications.

Rayner, Manny, Bouillon, Pierrette, Hockey, Beth Ann, & Chatzichrisafis, Nikos. 2006. REGULUS : A Generic Multilingual Open Source Platform for Grammar-Based Speech Applications. *Pages 783–788 of: Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC)*.

Rosner, Michael, Caruana, Joe, & Fabri, Ray. 1998. Maltilex: A computational lexicon for maltese. *In:* Rosner, Michael (ed), *Computational Approaches to Semitic Languages: Proceedings of the Workshop held at COLING-ACL98*.

Rosner, Michael, Fabri, Ray, Caruana, Joe, Loughraïeb, M, Montebello, Matthew, Galea, David, & Mangion, Gordon K. 1999. Linguistic and Computational Aspects of Maltilex. *Pages 2–10 of: In Arabic Translation and Localisation Symposium: Proceedings of the Workshop*.

Rosner, Michael, Fabri, Ray, Attard, Duncan Paul, & Gatt, Albert. 2006. MLRS, a Resource Server for the Maltese Language. *Pages 90–98 of:* Abela, John, Dalli, Angelo, & Guillaumier, Kristian (eds), *Computer Science Annual Workshop (CSAW)*. University of Malta.

Rosner, Mike, & Joachimsen, Jan. 2012. The Maltese Language in the Digital Age. *In:* Rehm, George, & Uszkoreit, Hans (eds), *META-NET White Paper Series*. Springer.

Sag, Ivan A., Boas, Hans C., & Kay, Paul. 2012. Introducing Sign-Based Construction Grammar. *Chap. 1 of:* Boas, Hans C., & Sag, Ivan A. (eds), *Sign-Based Construction Grammar*. CSLI Publications.

Schembri, Tamara. 2012. The Broken Plural in Maltese: A Description. *Il-Lingwa Tagħna*, **3**.

Serracino-Inglott, Mario. 2003. *Id-Dizzjunarju Malti*. 2nd edn. Blata l-Bajda, Malta: Merlin.

Spagnol, Michael. 2011. *A Tale of Two Morphologies: Verb structure and argument alternations in Maltese*. PhD Thesis, University of Konstanz.

Steedman, Mark, & Baldridge, Jason. 2005. Combinatory Categorial Grammar. *Pages 1–62 of:* Borsley, R., & Borjars, K. (eds), *Non-Transformational Syntax*. Blackwell.

Stolz, Thomas. 2011. Maltese. *Chap. 12, pages 241–256 of:* Kortmann, Bernd, & van der Auwera, Johan (eds), *The Languages and Linguistics of Europe: A Comprehensive Guide, Volume 1*. Berlin/Boston: De Gruyter Mouton.

Tiedemann, Jörg. 2009. News from OPUS-A collection of multilingual parallel corpora with tools and interfaces. *Pages 237–248 of:* Nicolov, N., Bontcheva, K., Angelova, G., & Mitkov, R. (eds), *Recent Advances in Natural Language Processing*. Amsterdam/Philadelphia: John Benjamins.

Vijay-Shanker, K., & Weir, DJ. 1994. The equivalence of four extensions of context-free grammars. *Mathematical systems theory*, **27**(6), 511–546.

Virk, Shafqat Mumtaz. 2013. *Computational Linguistics Resources for Indo-Iranian Languages*. Ph.D. Thesis, Chalmers University of Technology / University of Gothenburg.

Zammit, Angelo. 2012. *Extending the GF Framework towards Coverage of the Maltese Language*. Dissertation, University of Malta.