Verb Morphology of Hebrew and Maltese — Towards an Open Source Type Theoretical Resource Grammar in GF

Dana Dannélls* **and John J. Camilleri†**

*Department of Swedish Language, University of Gothenburg
SE-405 30 Gothenburg, Sweden
†Department of Intelligent Computer Systems, University of Malta
Msida MSD2080, Malta
dana.dannells@svenska.gu.se, jcam0003@um.edu.mt

Abstract

One of the first issues that a programmer must tackle when writing a complete computer program that processes natural language is how to design the morphological component. A typical morphological component should cover three main aspects in a given language: (1) the lexicon, i.e. how morphemes are encoded, (2) orthographic changes, and (3) morphotactic variations. This is in particular challenging when dealing with Semitic languages because of their non-concatenative morphology called root and pattern morphology. In this paper we describe the design of two morphological components for Hebrew and Maltese verbs in the context of the Grammatical Framework (GF). The components are implemented as a part of larger grammars and are currently under development. We found that although Hebrew and Maltese share some common characteristics in their morphology, it seems difficult to generalize morphosyntactic rules across Semitic verbs when the focus is towards computational linguistics motivated lexicons. We describe and compare the verb morphology of Hebrew and Maltese and motivate our implementation efforts towards a complete open source type theoretical resource grammars for Semitic languages. Future work will focus on semantic aspects of morphological processing.

1. Introduction

One of the first issues that a programmer must tackle when writing a complete computer program that processes natural language is how to design the morphological component. A typical morphological component should cover three main aspects in a given language: (1) the lexicon, i.e. how morphemes are encoded, (2) orthographic changes, and (3) morphotactic variations. This is in particular challenging when dealing with Semitic languages because of their non-concatenative morphology called root and pattern morphology (Goldberg, 1994).

The Grammatical Framework (GF) is a grammar formalism for multilingual grammars and their applications (Ranta, 2004). It has a Resource Grammar Library (Ranta, 2009) that is a set of parallel natural language grammars that can be used as a resource for various language processing tasks. Currently, the only Semitic morphological component included in the library is for Arabic (Dada and Ranta, 2007). To increase the coverage of Semitic languages we decided to develop two additional resource grammars for Hebrew and Maltese. The availability of several languages belonging to the same language family in one framework fosters the development of common language modules where grammatical rules across languages are generalised. Thus, increasing the potential of yielding interesting insights highlighting similarities and differences across languages. These kind of modules already exist in GF for Romance and Scandinavian languages.

In this paper we describe our implementations of Hebrew and Maltese verb morphologies in the context of GF. We present how two of the three morphological aspects mentioned above are accounted departing from the similarities and differences of verb formation in each of the two languages.

2. Verb morphology

Each of the Semitic languages has a set of verbal patterns, which is a sequence of vowels (and possibly consonants) into which root consonants are inserted. The root itself has no definite pronunciation until combined with a vocalic pattern, i.e. a template. The combination of morphological units is non-linear, i.e. it relies on intertwining between two independent morphemes (root and pattern).[1]

There are different ways in how templates modify the root consonants: doubling the middle consonants, inserting vowels between consonants, adding consonantal affixes, etc. Inflectional morphology systems are constructed by attaching prefixes and suffixes to lexemes. Verb lexemes are inflected for person, number, gender and tense. Common tenses of Semitic languages are: present, perfect, imperfect, and imperative.[2]

2.1. Modern Hebrew

Hebrew has seven verb pattern groups (binyanim) that are associated with a fixed morphological form, e.g. pa'al: C1aC2aC3, nif'al: niC1C2aC3, pi'el:C1iC2eC3. There are two major root classifications: regular (strong) and irregular (weak). In the same manner that each verb belongs to a particular binyan, it also belongs to a particular group of verbs (Hebrew gzarot) that classify them by their root composition (for an extensive information about the Hebrew root and pattern system see Arad (2005)). For regular verbs, all root consonants are present in all the verb forms, there are fixed rules that distinguish how verbs are

[1] Linguists consider the root to be a morpheme despite the fact that it is not a continuous element in the word, and it is not pronounceable (McCarthy, 1979; McCarthy, 1981).

[2] In Semitic languages, the past tense is referred by the term perfect and the future tense by imperfect.

conjugated depending on their guttural root letters, i.e. A, h, H, O.[3] An example of two verbs that are conjugated differently in pa'al future tense because in one of the verbs the root's second guttural letter is *h* are: Agmwr (g.m.r, 'will finish') with *w* stem vowel, and Anhag (n.h.g, 'will drive') with *a* stem vowel. Irregular verbs are verbs where one or more of the root consonants are either missing or altered, which causes some deviation from a fully regular conjugation. These verbs can be classified into three main groups, each of which contains three to five subgroups (Coffin and Bolozky, 2005). Roots are conjugated differently depending on the root classification group they belong to, e.g. ywred (y.r.d, pa'al, group2_py, 'he goes down'), yasen (y.s.n, pa'al, group3_py, 'he sleeps'). The sub-groups of irregular verbs contain large root composition variations which depend on the different occurrences of the root's consonants; phonological changes contribute to these irregularities in verbs forms and inflections.

The Hebrew binyanim are associated with a semantic trait. This leads to a certain complexity when designing a morphological component. What leads to this complexity is the fact that some morphemes (roots) are combined with more than one pattern, resulting in ambiguity problem. On the other hand not all roots are realised in all patterns. To avoid inefficient parsing that generates too many results, that in turn introduces new difficulties in identifying the root's consonants and in resolving ambiguities, it is necessary to employ semantic markings in the lexicon.

2.2. Maltese

Maltese verb pattern groups (themes) are a subset Classical Arabic pattern groups. These patterns involve affixation and prefixation, for example, niżżel (theme II, 'bring down'), tniżżel (theme V, 'be brought down'). Verbs in theme I must be specified as undergoing a vowel change which is always $a \rightarrow o$ or $e \rightarrow o$. Theme II is defined by double middle radical, the vowel possibilities are fixed. Most of the Semitic Maltese verbal themes exhibit the same properties that can be seen in theme I and II.

The vowels of the Semitic Maltese verb templates, unlike those of Classical Arabic, do not have a fixed vowel pattern, rather a vast range of vowel patterns. Each template allows several different vowel patterns determined by the tense and person of conjugation. For example, the root's template *ħ-d-m*, under perfect 3rd person singular, takes the pattern *a-a* (ħadem), whilst the past participle takes the pattern *i-a* (ħidma).

Each verb stem has two vowels and there are seven different verb types. Since a very large number of Maltese verbs are borrowed from Romance (Sicilian and Italian) and English, the productive verbal morphology is mainly affixal with a concatenative nature (Hoberman and Aronoff, 2003). The synchronic, productive processes of verb derivation, has resulted in three distinctive verb morphology features that are often referred in terms of: Semitic Maltese, Romance Maltese, and English Maltese (Mifsud, 1995).

Roots can be classified into one of five groups: strong, weak, defective, hollow, double and quadriliteral (4 radicals instead of 3). A root bears a semantic meaning that is converted into passive, active, reflexive forms depending on the pattern it belongs, e.g. *ħ-r-ġ* 'out', ħriġna 'we went out'.

Conjugations have predictable patterns and it is possible to predict the patterns and the entire conjugation tables from a given verb form (Aquilina, 1960; Aquilina, 1962). This may motivate the choice of representing lexemes in the lexicon (Ussishkin and Twist, 2007).

3. The Grammatical Framework (GF)

The Grammatical Framework is a functional grammar formalism based on Martin-Löf's type-theory (Martin-Löf, 1975) implemented in Haskell.

GF has three main module types: abstract, concrete, and resource. Abstract and concrete modules are top-level in the sense that they appear in grammars that are used at runtime for parsing and generation. One abstract grammar can have several corresponding concrete grammars; a concrete grammar specifies how the abstract grammar rules should be linearized in a compositional manner. A resource grammar is intended to define common parts of the concrete syntax in application grammars. It contains linguistic operations and parameters that are used to produce different forms and can be used as inherent features.

GF has a Resource Grammar Library, i.e. a set of parallel grammars that are built upon one abstract syntax. The GF's library, containing grammar rules for seventeen languages,[4] plays the role of a standard software library (Ranta, 2009). It is designed to gather and encapsulate morphological and syntactic rules of languages, which normally require expert knowledge, and make them available for non-expert application programmers by defining a complete set of morphological paradigms and a syntax for each language.

4. The grammar design

In this section we present how the verb morphologies of Modern Hebrew and Maltese are implemented in GF. The presented code fragments do not cover all aspect of the verb, such as passive/active mood, Hebrew infinitive form, Hebrew verbs with obligatory prepositions, English Maltese, etc. However, the code provides a glimpse of the two computational resources that are being developed. The presented code contains parameters, operations and lexicon linearizations which are defined according to GF's concrete and resource syntaxes. Parameters are defined to deal with agreement, operations are functions that form inflection tables, linearizations are string realisations of functions that are defined in the abstract syntax.

4.1. Common parameters

Both languages share the same parameter types and attributes for verbs, including: number (Singular, Plural),

[3] Throughout the paper we regulate the encoding of Hebrew characters using ISO-8859-8.

[4] The Resource Grammar Library currently (2010) contains the 17 languages: Arabic (complete morphology), Bulgarian, Catalan, Danish, Dutch, English, Finnish, French, German, Italian, Norwegian (bokmål), Polish, Romanian, Russian, Spanish, Swedish and Urdu.

gender (Masculine, Feminine), case (Nominative, Accusative, Genitive), person (first, second, third), voice (Active, Passive) and tense (Perfect, Participle, Imperfect). These types have the following definitions in GF syntax:

```
Number  = Sg | Pl ;
Gender  = Masc | Fem ;
Case    = Nom | Acc | Gen ;
Person  = P1 | P2 | P3 ;
Voice   = Active | Passive ;
Tense   = Perf | Part | Imperf ;
```

4.2. Modern Hebrew

An additional parameter *VPersonNumGen* provides a detailed description about how verbs are inflected. The parameter's attributes indicate: first person singular/plural, second and third person singular/plural and gender.

```
VPerNumGen = Vp1Sg | Vp1Pl | Vp2Sg Gender
  | Vp2Pl Gender | Vp3Sg Gender
  | Vp3Pl Gender ;
```

Operations

The Hebrew operations include: *Pattern*, i.e. a string consisting of a four position pattern slot, *Root*, i.e. a string consisting of either three or four (*Root4*) consonants. The Hebrew *Verb* is defined as a string that is inflected for tense person, number and gender. The *mkVPaal* operation defines regular verb paradigms for each tense and agreement features. The operation *getRoot* associates every consonant in the input string *v* with a variable. This is accomplished by the operation *C@?* which binds each consonant in the string *s* to a variable, e.g. C1 and C2. These variables are than coded into patterns using the operation *appPattern* which specifies how the root's consonants should be inserted into a pattern, given a root and a pattern.

```
Pattern : Type={C1, C1C2, C2C3, C3 : Str};
Root    : Type={C1,C2,C3 : Str};
Root4   : Type=Root ** {C4 : Str};
Verb    : Type={s : Tense ⇒ VPerNumGen ⇒ Str };

mkVPaal : Str → Verb = \v →
let root = getRoot v
in {s = table {
  Perf ⇒ table {
    Vp1Sg ⇒ appPattern root C1aC2aC3ty ;
    Vp1Pl ⇒ appPattern root C1aC2aC3nw ;
    Vp2Sg Masc ⇒ appPattern root C1aC2aC3th ;
    Vp2Sg Fem  ⇒ appPattern root
        C1aC2aC3t ;
    Vp2Pl Masc ⇒ appPattern root C1aC2aC3tM ;
    ... }
  Imperf ⇒ table { ... }
  }
};

getRoot : Str → Root = \s → case s of {
  C1@? + C2@? + C3 =>
    {C1 = C1 ; C2 = C2 ; C3 = C3}
};

appPattern : Root → Pattern → Str = \r,p →
    p.C1 + r.C1 + p.C1C2 + r.C2 + p.C2C3 + r.C3 +
         p.C3 ;
```

Patterns

Root patterns are defined in a separate resource. Patterns specify consonant slots and morphological forms, some examples are:

```
C1aC2aC3ty = {C1=""; C1C2=""; C2C3=""; C3="ty"};
C1aC2aC3nw = {C1=""; C1C2=""; C2C3=""; C3="nw"};
C1aC2aC3th = {C1=""; C1C2=""; C2C3=""; C3="th"};
```

Lexicon

Lexicon entries are functions that are defined in the abstract syntax. Below is an example of how the three verb entries: *write_V2*, *pray_V* and *sleep_V*, are linearized in the Hebrew lexicon. The lexicon generates verb paradigms through their binyanim, using the Hebrew operations.

```
write_V2 = mkVPaal "ktb" ;
pray_V   = mkVHitpael "pll" ;
sleep_V  = mkVPaalGroup3_py "ysn";
```

4.3. Maltese

There are additional parameters defined for Maltese, these include: VerbType (Strong, Defective, Weak, Hollow, Double), VOrigin (Semitic, Romance), VForm (for possible tenses, persons and numbers).

```
VType   = Strong | Defective | Weak | Hollow |
    Double ;
VOrigin = Semitic | Romance ;
VForm   = VPerf PerGenNum | VImpf PerGenNum |
    VImp Number ;
```

Operations

The operations for Maltese include: *Pattern*, i.e. a string consisting of two vowels, *Root*, i.e. a string consisting of four consonants of which one can be eliminated. The Maltese *Verb* is defined as a string inflected for tense, person, gender and number, that has the parameter values: *VerbType* and *VerbOrigin*. The *mkVerb* operation utilizes additional operations such as *classifyVerb, mkDefective, mkStrong* etc. to identify the correct verb. The operation *classifyVerb* takes a verb string and returns its root, pattern, and verb type, i.e. *Strong, Defective, Quad* etc. The operation *v1@#Vowel* matches the pattern *Vowel* and binds the variable *v1* to it. It is based on pattern matching of vowels.

```
Pattern : Type = {v1, v2 : Str} ;
Root : Type = {K, T, B, L : Str} ;
Verb : Type = {s : VForm ⇒ Str ; t : VType ; o :
     VOrigin} ;

mkVerb : Str → Verb = \mamma →
  let
    class = classifyVerb mamma
  in
    case class.t of {
      Strong ⇒ mkStrong class.r class.p ;
      Defective ⇒ mkDefective class.r class.p ;
      Quad ⇒ mkQuad class.r class.p ;
      ...
    } ;

classifyVerb : Str → { t:VType ; r:Root ;
  p:Pattern } = \mamma → case mamma of {
      K@#Consonant + v1@#Vowel
      + T@#Consonant + B@#Consonant
      + v2@#Vowel + L@#Consonant ⇒
```

```
      { t=Quad ; r={ K=K ; T=T ; B=B ; L=L } ;
      p={ v1=v1 ; v2=v2 } } ;
}
```

Lexicon

In this example, functions are linearized by using two different operations defined for: regular inflection of verbs (used in *write_V2*), where the verb is given in perfect tense, third person, singular, masculine and irregular inflection of verbs (used in *pray_V*), where two additional strings are given, namely the imperative singular and the imperative plural forms of the verb.

```
write_V2 = mkVerb "kiteb" ;
pray_V = mkVerb "talab" "itlob" "itolbu";
```

4.4. Inflection paradigm

An example of the output produced by GF for the verb 'write' is illustrated in Table 1.

Hebrew	Maltese
mkVPaal "ktb"	mkVerb "kiteb"
Perfect	
Vp1Sg ⇒ "ktbty"	(Per1 Sg) ⇒ "ktibt"
Vp1Pl ⇒ "ktbnw"	(Per1 Pl) ⇒"ktibna"
Vp2SgMasc ⇒ "ktbt"	(Per2 Sg) ⇒"ktibt"
Vp2SgFem ⇒ "ktbt"	
Vp2PlMasc ⇒ "ktbtM"	(Per2 Pl) ⇒"ktibtu"
Vp2PlFem ⇒ "ktbtN"	
Vp3SgMasc ⇒ "ktb"	(Per3Sg Masc) ⇒ "kiteb"
Vp3SgFem ⇒ "ktbh"	(Per3Sg Fem) ⇒ "kitbet"
Vp3PlMasc ⇒ "ktbw"	Per3Pl ⇒ "kitbu"
Vp3PlFem ⇒ "ktbw"	
Imperfect	
Vp1Sg ⇒ "Aktwb"	(Per1 Sg) ⇒ "nikteb"
Vp1Pl ⇒ "nktwb"	(Per1 Pl) ⇒ "niktbu"
Vp2SgMasc ⇒ "tktwb"	(Per2 Sg) ⇒ "tikteb"
Vp2SgFem ⇒ "tktby"	
Vp2PlMasc ⇒ "tktbw"	(Per2 Pl) ⇒ "tiktbu"
Vp2PlFem ⇒ "tktbw"	
Vp3SgMasc ⇒ "yktwb"	(Per3Sg Masc) ⇒ "jikteb"
Vp3SgFem ⇒ "tktwb"	(Per3Sg Fem) ⇒ "tikteb"
Vp3PlMasc ⇒ "yktbw"	Per3Pl ⇒ "jiktbu"
Vp3PlFem ⇒ "yktbw"	

Table 1: Example of Hebrew and Maltese verb inflection tables of the verb 'write'.

5. State of the work

The core syntax implemented for the two languages has around 13 categories and 22 construction functions. It covers simple syntactic constructions including predication rules which are built from noun and verb phrases.

The lexicons were manually populated with a small number of lexical units, covering around 20 verbs and 10 nouns in each language. The Maltese verb morphology covers the root groups: strong, defective and quadriliteral. In Hebrew, the strong verb paradigms and five weak verb paradigms in binyan pa'al are covered.

6. Discussion and related work

Although there are already some morphological analyzers available for Hebrew (Itai and Wintner, 2008; Yona and Wintner., 2008) and data resources available for Maltese (Rosner et al., 1999), they are not directly usable within the Grammatical Framework. To exploit the advantages offered by GF, the language's grammar must be implemented in this formalism. One of the advantages of implementing Semitic non-concatenative morphology in a typed language such as GF compared with other finite state languages is that strings are formed by records, and not through concatenation. Moreover, once the core grammar is defined and the structure and the form of the lexicon is determined, it is possible to automatically acquire lexical entries from exiting lexical resources. In the context of GF, three wide-coverage lexicons have been acquired automatically: Bulgarian (Angelov, 2008b), Finnish (Tutkimuskeskus, 2006) and Swedish (Angelov, 2008a).

In this work, the design decisions taken by the programmers are based on different points of arguments concerning the division of labour between a linguistically trained grammarian and a lexicographer. The Maltese implementation consider stems in the lexicon rather than patterns and roots, cf. Rosner et al. (1998); in the framework of GF, classes of inflectional phenomena are given an abstract representation that interact with the root and pattern system. In Hebrew, recognizing prefixes and suffixes is not always sufficient for recognizing the root of the verb. Although root recognition is mandatory for generating the verb's complete conjugation table, changes in patterns and the absence of root letters in different lexemes make it increasingly hard to infer the root (Deutsch and Frost, 2002) which requires a large amount of tri-consonantal constraints. This is in particular true for lexemes derived from weak roots where one of the root consonants is often missing (Frost et al., 2000). To avoid a large amount of morphosyntactic rules, we choose to employ semantic markings in the lexicon by specifying roots and patterns instead of lexemes; this computationally motivated approach becomes plausible since the meaning of the lexeme is already known.

7. Conclusions and Future Work

In this paper we have presented implementations of Hebrew and Maltese components that tend to convey the non-concatenative morphology of their verbs. Although we could identify common characteristics among these two Semitic languages, we found it difficult to generalize morphosyntactic rules across Semitic verbs when the focus is towards a computational motivated lexicon.

When designing a computer system that can process several languages automatically it is useful to generalize as many morphosyntactic rules across languages that belong to the same language group. One fundamental question that rises from our implementations is to what extent we can generalize the concrete syntaxes of Semitic languages. One way to approach this question is by employing semantic markings in the lexicons of the Semitic languages and focus on semantic aspects of morphological processing. This remains for future work.

8. References

Krasimir Angelov. 2008a. Importing SALDO in GF. http://spraakbanken.gu.se/personal/lars/kurs/lgres08/tpaper/angelov.pdf.

Krasimir Angelov. 2008b. Type-theoretical Bulgarian grammar. In In B. Nordström and eds. A. Ranta, editors, *Advances in Natural Language Processing (GoTAL 2008)*, volume 5221 of *LNCS/LNAI*, pages 52—-64.

Joseph Aquilina. 1960. *The structure of Maltese*. Valletta: Royal University of Malta.

Joseph Aquilina. 1962. *Papers in Maltese linguistics*. Royal University, Malta.

Maya Arad. 2005. *Roots and patterns: Hebrew morpho-syntax*. Springer, The Netherlands.

Edna Amir Coffin and Shmuel Bolozky. 2005. *A Reference Grammar of Modern Hebrew*. Cambridge University Press.

Ali Dada and Aarne Ranta. 2007. Implementing an open source arabic resource grammar in GF. In M. Mughazy, editor, *Perspectives on Arabic Linguistics.*, Papers from the Twentieth Annual Symposium on Arabic Linguistics. John Benjamins Publishing Company, March 26.

Avital Deutsch and Ram Frost, 2002. *Lexical organization and lexical access in a non-concatenated morphology*, chapter 9. John Benjamins.

R. Frost, A. Deutsch, and K. I. Forster. 2000. Decomposing morphologically complex words in a nonlinear morphology. *Journal of Experimental Psychology: Learning, Memory and Cognition*, 26(3):751–765.

Gideon Goldberg. 1994. Principles of Semitic word-structure. In G. Goldberg and S. Raz, editors, *In Semantic and ushitic studies*, pages 29–64. Wiesbaden:Harrassowitz.

Robert D. Hoberman and M. Aronoff, 2003. *The verbal morphology of Maltese: From Semitic to Romance*, chapter 3, pages 61–78. Amsterdam: John Benjamins.

Alon Itai and Shuly Wintner. 2008. Language resources for Hebrew. *Language Resources and Evaluation*, 42(1):75–98, March.

P. Martin-Löf. 1975. An intuitionistic theory of types: Predicative part. In H. E. Rose and J. C. Shepherdson, editors, *Proc. of Logic Colloquium '73, Bristol, UK*, volume 80, pages 73–118. North-Holland.

John J. McCarthy. 1979. On stress and syllabification. *Linguistic Inquiry*, 10:443–465.

John J. McCarthy. 1981. The representation of consonant length in Hebrew. *Linguistic Inquiry*, 12:322–327.

Manwel Mifsud. 1995. *Loan verbs in Maltese: a descriptive and comparative study*. Leiden, New York, USA.

Aarne Ranta. 2004. Grammatical framework, a type-theoretical grammar formalism. *Journal of Functional Programming*, 14(2):145–189.

Aarne Ranta. 2009. The GF resource grammar library. *The on-line journal Linguistics in Language Technology (LiLT)*, 2(2). http://elanguage.net/journals/index.php/lilt/article/viewFile/214/158.

M. Rosner, J. Caruana R., and Fabri. 1998. Maltilex: a computational lexicon for Maltese. In *Semitic '98: Proceedings of the Workshop on Computational Approaches to Semitic Languages*, pages 97–101, Morristown, NJ, USA. Association for Computational Linguistics.

M. Rosner, J. Caruana, and R. Fabri. 1999. Linguistic and computational aspects of maltilex. In *Arabic Translation and Localisation Symposium: Proceedings of the Workshop*, pages 2—-10, Tunis.

Kotimaisten Kielten Tutkimuskeskus. 2006. KOTUS wordlist. http://kaino.kotus.fi/sanat/nykysuomi.

Adam Ussishkin and Alina Twist. 2007. Lexical access in Maltese using visual and auditory lexical decision. In *Conference of Maltese Linguistics*, L-Ghaqda Internazzjonali tal-Lingwistika Maltija, University of Bremen, Germany, October.

Shlomo Yona and Shuly Wintner. 2008. A finite-state morphological grammar of Hebrew. *Natural Language Engineering*, 14(2):173–190, April.