Ali AlDada, Aarne Ranta
Department of Computing Science
Chalmers University of Technology and Göteborg Universitet - Suede
eldada@student.chalmers.se, aarne@cs.chalmers.se

Open Source Arabic Grammars in Grammatical Framework

1. Introduction

Language technology and software localization consume a significant share of many companies' time and work. Translating an operating system or an application to dif- ferent languages requires a language expert for each new language, and will still involve language-related problems. We illustrate this difficulty with an example. A mail reader application wants to display messages like

- You have 1 new message
- You have 2 new messages
- You have 3 new messages
- You have 100 new messages

If these are to be translated into Arabic, special mor- phological and syntactic considerations should be made, which include inflecting "message" in number:

- 1 message éثAƒP risālatun
- 2 messages àAJثAƒP risālatāni
- (3-10) messages ةKAƒP rasā-ila
- (11-99) messages éثAƒP risālatan
- x100 messages éثAƒP ris¯alatin

So the word "messages" is translated into five different words in Arabic, depending on its number and case. This is not to mention noun-adjective agreement which should be taken care of when translating "new messages" into

Arabic. We review in this paper a solution based on devel-

oping libraries of natural language constructs and rules, which can be used by an application programmer who is not knowledgeable in a specific language. The program- ming language we use to develop the libraries is Gram- matical Framework (GF) [Ranta, 2004], a special-purpose functional programming language. The language library, called a resource grammar [Khegai and Ranta, 2004], con- sists of rules that span a language's orthography, mor- phology, and syntax. This library can be reused in ap- plications through an Application Programming Interface

(API) by programmers that are unaware of the details of the specific natural language. We hope that Arabic language technology can benefit from this grammar reuse principle, especially that the Resource Grammar itself is available as Open Source [Ranta, 2006].

We have accomplished significant work implementing parts of the Arabic orthography, morphology, and syn- tax, along with providing a sample lexicon based on the Swadesh list [Hymes, 1960]. We give below brief examples

from both the morphology and the syntax.

2 Morphology Example

Arabic morphotactics can be described in terms of tricon- sonantal roots and CV patterns. With the help of GF's record types, we define types for roots and patterns as follows:

```
Root : Type = {f, c, l: Str} ;
Pattern : Type = {h, m1, m2, t : Str};
```

where f, c, and l stand for the three root consonants, and h, m1, m2, and t are the pattern constituents. We also define functions to interdigitize roots and patterns:

```
interdigitize : Root -> Pattern -> Str =
\fcl, p ->
p.h + fcl.f + p.m1 + fcl.c + p.m2 + fcl.l + p.t;
```

Below we give an example of a GF table that gives a feel for how the GF formalism parallels the way grammarians think of languages. The example we give is the table of suffixes of the Arabic (active and passive voice) perfect tense, which is one of the first tables given in an Arabic grammar book.

```
perfectSuffix :  PerGenNum => Str = table {
   Per3 Masc Sg => "a";          كَتَبَ  kataba
   Per3 Masc Dl => "aA";         كَتَبَا  katabā
   Per3 Masc Pl => "UA";         كَتَبُوا  katabuw
   Per3 Fem  Sg => "ato";        كَتَبَت  katabat
   Per3 Fem  Dl => "ataA";       كَتَبَتَا  katabatā
   Per3 Fem  Pl => "ona";        كَتَبْنَ  katabna
   Per2 Masc Sg => "ota";        كَتَبْتَ  katabta
   Per2 _    Dl => "otumaA";     كَتَبْتُمَا  katabtumā
   Per2 Masc Pl => "otumo";      كَتَبْتُمْ  katabtum
   Per2 Fem  Sg => "oti";        كَتَبْتِ  katabti
   Per2 Fem  Pl => "otun~a";     كَتَبْتُنَّ  katabtunna
   Per1 Sing    => "otu";        كَتَبْتُ  katabtu
   Per1 Plur    => "onaA"        كَتَبْنَا  katabnā
```

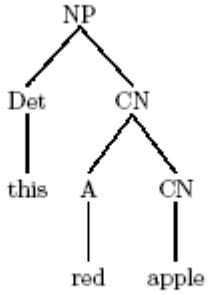

Figure 1: Abstract syntax tree for the example Noun Phrase.

The type of CN shows that an Arabic noun has an inherent gender and species, but is inflected for number, state, and case. A simplified version of rule (1) can be implemented as shown below.

```
DetCN det cn = {
s = \\c =>
det.s ! cn.h ! cn.g ! c
++ cn.s ! det.n ! det.d ! c ;
red
apple
n = det.n;
g = cn.g;
```

Figure 1: Abstract syntax tree for the example Noun Phrase.

3 Syntax Example

We take as an example translating the following Noun Phrase, whose abstract syntax tree is shown in figure 1, from English into Arabic:

هٰذِهِ التُّفّاحَة الحَمْراء
hādihi 't-tuffāḥah al-ḥamrā'
this red apple

¯Our goal is that a user can get the Arabic concrete rep- resentation from the abstract description. This requires that each branch of the tree, which describes a syntactic rule, be implemented for Arabic. We will give an example that shows this.

The syntactic rule that takes a determiner (Det) and a common noun (CN) to produce a noun phrase (NP) has the following type signature:

DetCN : Det -> CN -> NP; (1)

DetCN is the name of the rule, Det and CN are the types of the two arguments, and NP is the type of the value. The NP (Z@QÌmÌ'@ ékA®JË@ èYë hādihi 't-tuffāḥah al-ḥamrā-) should p = P3 } ;

The rule accepts a case c from the sentence context using the table construction operator \\. The rule then says that the determiner's string is chosen from its inflection table given the NP case and the CN's gender and species. Similarly, the noun's string is chosen using the NP case and the determiner's number and definiteness. Both strings are then concatenated together. This is just the formal way to define what we descriptively explained with the Z@QÌmÌ'@ ékA®JË@ èYë hādihi 't-tuffāḥah al-ḥamrā- example. The user requires,

in addition to the morphological and syntactic correctness, lexical entries for the words she will use. Thus, an Arabic application programmer should only need to make sure that the necessary Arabic lexical entries are available in the lexicon. If not, these entries can be added as follows (we also give a lexical entry of the verb "to eat"):

apple_N = sdfN "tfH" "fuc~Alp" Fem NoHum ;

red_A = clrA "Hmr" ;

eat_V = v1 "?kl" a u ;

where the three-letter strings are the roots, fuc~Alp is a CV pattern, and the two entries after "?kl" are the vowels associated with the perfect and imperfect tenses in Arabic verbs. This way of specifying verbs in the lexicon (root,

be properly construc
ted, so for example the determiner's

verb form, and necessary vowels) is consistent with the gender should be derived from that of the noun; since the noun ékA®K tuffāḥah is feminine, the determiner should be in its feminine form (èYë hādihi and not @Yë hadā). Sim-

and definiteness should follow

way how most Arabic dictionaries are written.

4 Related Work

ilarly the noun's number

that of the determiner, so we inflect Z@QÔg ékA®K tuffāḥah ḥamrā- as singular definite because èYë hādihi is singular

A large-scale implementation of the Arabic morphologi-

and definite.

cal system is the Xerox Arabic Morphological Analyzer

We give types to each word category specifying how its inflection table looks like and which are its inherent features. For our example categories we define the types as follows:

Det = {s : Species => Gender => Case => Str;

n : Number; d : State } ;

CN = {s : Number => State => Case => Str;

g : Gender; h : Species } ;

NP = {s : Case => Str ;

n : Number; g : Gender; p : Person } ;

and Generator [Beesley, 1996, 1998, Beesley and Kart- tunen, 2000, Beesley, 2001]. This system is developed using only the Xerox Finite State Technology tools [Beesley and Karttunen, 2003] from which an Arabic Finite State Lexical Transducer is written.

Other notable computational models of the Arabic mor- phology are Tim Buckwalter's Arabic Morphological Ana- lyzer [Buckwalter, 2002, 2004b,a] and Kiraz's multi-tape two-level formalism for the root-and-pattern morphology of Semitic languages [Kiraz, 1994, 1995].

We give in the paper details about these and other Ara- bic systems, pointing out their merits and approaches. We evaluate also our own system, noting than GF has lots

of merits and strengths when describing linguistic issues, such as inflection tables and grammar rules. The ability to use GF with a "different" language like Arabic was a prerequisite for the whole work. We illustrate by exam- ples the elegance in which a grammar formalism like GF enables us to represent linguistic rules and abstractions.

References

1. Sam Ammar and Joseph Dichy. Bescherelle Les verbes arabes. Hatier, Paris, France, 1999.
2. Kenneth R Beesley. Finite-State Morphological Analysis and Generation of Arabic at Xerox Research: Status and Plans in 2001. In Workshop Proceedings on Arabic Language Processing: Status and Prospects, pages 1–8, Toulouse, 2001. ACL.
3. Kenneth R Beesley. Arabic Finite-State Morphological Analysis and Generation. In COLING96, pages 89–94, Copenhagen, 1996. Center for Sprogteknologi.
4. Kenneth R Beesley. Arabic morphology using only finite- state operations. In Proceedings of the Workshop on Computational Approaches to Semitic Languages, pages 50–57, 1998.
5. Kenneth R Beesley and Lauri Karttunen. Finite-state non-concatenative morphotactics. In Proceedings of the Fifth Workshop of the ACL Special Interest Group in Computational Phonology, pages 1–12, 2000.
6. Kenneth R. Beesley and Lauri Karttunen. Finite State Morphology. CSLI Studies in Computational Linguis- tics. CSLI Publications, Stanford, California, 2003.
7. Tim Buckwalter. Issues in Arabic Orthography and Morphology Analysis. In Proceedings of the COLING Workshop on Computational Approaches to Ara- bic Script-based Languages, pages 31–34, 2004a.
8. Tim Buckwalter. Buckwalter Arabic Morphological Ana- lyzer Version 1.0. LDC catalog number LDC2002L49, ISBN 1-58563-257-0, 2002.
9. Tim Buckwalter. Buckwalter Arabic Morphological Ana- lyzer Version 2.0. LDC catalog number LDC2004L02, ISBN 1-58563-324-0, 2004b.
10. Antoine El-Dahdah. A Dictionary of Arabic Verb Con- jugataion. Librairie du Liban, Beirut, Lebanon, fourth edition, 1999.
11. Markus Forsberg and Aarne Ranta. Functional Morphol- ogy. In Proceedings of the Ninth ACM SIGPLAN Inter- national Conference on Functional Programming, ICFP 2004, pages 213–223. ACM Press, 2004.
12. D. H. Hymes. Lexicostatistics so far. Current Anthropol- ogy, 1:3–44, 1960.
13. Janna Khegai and Aarne Ranta. Building and Using a Russian Resource Grammar in GF. In Intelligent Text Processing and Computational Linguistics (CICLing- 2004), pages 38–41, Seoul, Korea, 2004.
14. George A Kiraz. Computational Nonlinear Morphology with Emphasis on Semitic Languages. Studies in Natu- ral Language Processing. Cambridge University Press, 2001.
15. George A Kiraz. Multi-Tape Two-Level morphology: a case study in Semitic Non-Linear morphology. In COL- ING 94: Proceedings of the 15th International Confer- ence on Computational Linguistics, volume 1, Kyoto, Japan, 1994.

16. George A Kiraz. Computational Analy- ses of Arabic Morphology, 1995. URL http://www.bell-labs.com/project/tts/gkiraz- arabmorph.ps.

17. Aarne Ranta. Grammatical Framework: A Type- theoretical Grammar Formalism. Journal of Functional Programming, 14:145–189, 2004.

18. Aarne Ranta. GF Resource Grammar Library v. 1.0, 2006. URL http://www.cs.chalmers.se/~aarne/GF/lib/resource-1.0/doc/. Otakar Smrž. Functional Arabic Morphology. Formal Sys- tem and Implementation. PhD thesis, Charles University in Prague, in prep.