IMORPHĒ: An Inheritance and Equivalence Based Morphology Description Compiler

Violetta Cavalli-Sforza, Abdelhadi Soudi

Language Technologies Institute, Carnegie Mellon University
5000 Forbes Avenue, Pittsburgh, PA 15217, U.S.A.
violetta@cs.cmu.edu

Center for Computational Linguistics, Ecole Nationale de l'Industrie Minérale
Avenue Hadj Ahmed Cherkaoui, B.P. 753, Agdal, Rabat, Morocco
asoudi@enim.ac.ma, asoudi2002@yahoo.fr

Abstract

IMORPHĒ is a significantly extended version of MORPHĒ, a morphology description compiler. MORPHĒ's morphology description language is based on two constructs: 1) a morphological form hierarchy, whose nodes relate and differentiate surface forms in terms of the common and distinguishing inflectional features of lexical items; and 2) transformational rules, attached to leaf nodes of the hierarchy, which generate the surface form of an item from the base form stored in the lexicon. While MORPHĒ's approach to morphology description is intuitively appealing and was successfully used for generating the morphology of several European languages, its application to Modern Standard Arabic yielded morphological descriptions that were highly complex and redundant. Previous modifications and enhancements attempted to capture more elegantly and concisely different aspects of the complex morphology of Arabic, finding theoretical grounding in Lexeme-Based Morphology. Those extensions are being incorporated in a more flexible and less *ad hoc* fashion in IMORPHĒ, which retains the unique features of our previous work but embeds them in an inheritance-based framework in order to achieve even more concise and modular morphology descriptions and greater runtime efficiency, and lays the groundwork for IMORPHĒ to become an analyzer as well as a generator.[1]

1. Introduction

Computational models of morphology, particularly for languages with complex morphology (e.g., Semitic language root-and-pattern morphology and Amerindian language polysynthetic morphology) have attracted significant attention from the computational linguistics community over the last two decades. Morphological analyzers and generators play an important role in many natural language processing technologies and applications, ranging from monolingual and cross-lingual information retrieval to natural language generation, from machine translation to computer-assisted language learning.

Several different approaches to computational morphology have been proposed. The MORPHIX system (Finkler and Neumann, 1988) used a classification-based approach to German morphological analysis, and combined trie data structures with finite-state technology. Starting with the general two-level morphology model (Koskenniemi, 1983), a number of other approaches based on finite-state transducers have specifically addressed the morphology of Semitic languages in general, and Arabic in particular (Beesley et al., 2989; Beesley, 1996; Kiraz, 1998, 2000). These approaches are formally elegant, computationally efficient, and inherently bidirectional, but they can be quite difficult to build and maintain. Characteristically, they give equal status to different components of the word (root, pattern, prefixes and suffixes), a claim that is contested by some linguists. For Modern Standard Arabic, at least, the dictionary-morphology connection is somewhat problematic, since the mapping between root-plus-pattern and meaning is not as systematic as one might wish and meanings are more reliably associated with stems.

More recently, researchers on Modern Standard Arabic morphology have argued that stem-grounded lexical databases, whose entries are associated with grammar and lexis specifications, are the most appropriate way of organizing and storing pertinent information for Arabic (Dichy & Farghaly, 2003; Cavalli-Sforza & Soudi, forthcoming). Indeed, the most commonly used Arabic morphological analyzer (Buckwalter, 2004) is stem-based, though it still treats inflectional and other prefixes and suffixes as equal-status items, relying on dictionaries of prefixes, stems and suffixes and on compatibility tables to perform analysis. The Buckwalter analyzer has also been re-engineered for use in generation (Habash, 2004).

Departing from the knowledge-based tradition of hand-coded rules and lexicons, which also includes Prolog-rule based approaches (e.g., Macks, 2002), several authors have developed systems that learn Semitic morphology from corpora (Darwish, 2003; Itai & Segal, 2003; Marsi et al., 2005), possibly aided by user feedback (Abuleil et al., 2002). Since (semi-)automatic learning of morphology is a practical and quick way of acquiring morphological knowledge (modulo tagging work required for training), empirically-based approaches are a promising for acquiring new morphological knowledge.

Our work on IMORPHĒ is grounded in the knowledge-based tradition, in that it requires a hand-coded morphology description, but the description is written in a simple and intuitively appealing formalism consisting of two components. The first is a hierarchy of morphological forms defined by the intrinsic features of lexical items and the additional features attributed (in generation) or extracted (in analysis) by a client, features that are common to groups of forms or distinguish forms from one another. The second is a set of transformational rules. In generation, which has been our focus to date, the

[1] Work on EMORPHĒ and IMORPHĒ has been partially supported by grant OISE-0107369 from the National Science Foundation.

condition part of the rule performs regular expression matching and the action part modifies the dictionary form of a lexical item. This much was already present in MORPHĒ, the tool that our work is based on. In order to reduce the bushiness of the hierarchy and minimize redundancy in a morphological descriptions, as well as to make explicit the similarity of different morphological forms across distant portions of the hierarchy, we augmented MORPHĒ with a number of linguistically and practically motivated enhancements. Most recently, we have been casting those enhancements in an inheritance-based framework, as is naturally suggested by the use of a morphological form hierarchy. Although differing in formalism, search strategy and implementation, our work bears a clear relationship to the approach taken in DATR/KATR (Finkel & Stump, 2002) to generating Hebrew verb morphology. While much of the impetus for developing IMORPHĒ has come from using MORPHĒ with Modern Standard Arabic, the framework itself is completely language independent and its earlier and simpler versions have indeed been applied to several other languages including English, French, German, Italian, Portuguese, Russian, and Spanish.

In the remainder of this paper, we take a historical approach to IMORPHĒ's development. We begin by describing the original MORPHĒ system and some of the extensions included in EMORPHĒ (Cavalli-Sforza & Soudi, 1993), motivating those extensions with a brief overview of Arabic morphology. We then discuss the limitations of EMORPHĒ, how those were addressed in IMORPHĒ, and provide some details about the specific differences between the two systems. We conclude with some remarks about IMORPHĒ's application to Modern Standard Arabic, present status and future work.

2. The Original MORPHĒ Compiler

IMORPHĒ's approach to morphology description is drawn from the original MORPHĒ Morphological Rule Compiler (Leavitt, 1994), developed at Carnegie Mellon University in the mid 90's, and from more recent linguistically–motivated enhancements designed to capture, in a concise and elegant way, the structure of words in languages with complex morphologies. MORPHĒ was developed in the context of the KANT machine translation project (Nyberg & Mitamura, 1992) and was used in translation from English to several target languages. The much simpler morphology of English, was analyzed with a different tool, so the analysis capability of the MORPHĒ was never fully implemented. At present, MORPHĒ and its successors still function only as generators of surface morphological forms; analysis is left for future work.

MORPHĒ's approach is centered on the concept of a Morphological Hierarchy Form (MFH). The MFH relates and contrasts different morphological forms – the leaf nodes of the hierarchy – by specifying, for each node on the path from the root to a leaf, the combination of features and values that distinguish a node from its parent and siblings. Transformational rules, attached to the leaf nodes, add, remove and replace prefixes, suffixes and infixes, and perform regular substring mappings that result in the surface form of a word. A morphology description for a language consists of the specification of the MFH and a set of transformational rules. MORPHĒ compiles the morphology description into Common Lisp functions that are themselves optionally compiled into object code for faster runtime performance.

In generation, MORPHĒ takes as its runtime input a feature structure (FS) that representing the lexical item that MORPHĒ must transform. The FS includes the base (dictionary) form of the item, its *intrinsic* (*static*) features, obtained from the lexicon, and *extrinsic* (*dynamic*) features obtained from the client, for example a sentence generator or a machine translation system. The base form is identified by a specific feature name. The input is pushed down a path in the MFH, which acts as a discrimination net, by matching feature-value pairs in the MFH and the FS at each node. When a leaf node is reached, transformational rules are applied to the base form of the input to produce a string output.

3. Extending MORPHĒ for Arabic

While MORPHĒ's MFH-plus-transformational rules approach is a conceptually attractive way of organizing a morphology description, the limitations of the morphology description language in the original MORPHĒ system proved unwieldy when applied to languages with complex root-and-pattern morphology such as Modern Standard Arabic (MSA). While a full description of Arabic morphology is beyond the scope of this paper, we highlight the major features that are relevant to the development of IMORPHĒ and its predecessors. Further details are provided in our previous work (Cavalli-Sforza & Soudi, 2003, forthcoming; Soudi et al., 2002, 2001, 2002; Cavalli-Sforza et al., 2000).

3.1. Arabic Morphology

The parts of speech that are inflected in MSA are verbs, nouns and adjectives. Additional bound morphemes that can also occur as prefixes or suffixes include: conjunctions, prepositions, vocative, exclamation and question particles and the future particle "sa" as prefixes; possessive and direct object pronouns as suffixes. The definite article "Al" is always attached to and precedes the noun or adjective, and loses the "A" with the preposition "li". The morphology of Arabic dialects is similar to but simpler than that of MSA. In what follows we focus on the inflectional morphology of MSA.

3.1.1. Arabic Verb Morphology

The Arabic verbal system is very rich in forms. Arabic verbs are based on three or four radicals, the letters that constitute the skeleton of the verb. An Arabic verb can be conjugated according to one of the traditionally recognized patterns: 15 triliteral patterns (with 3 radicals), of which at least 9 are in common use, and 4 less common quadriliteral patterns with 4 radicals), some quite rare. Verb patterns are derived by composing the verb root with different vowel and consonant patterns. Within each pattern, verbs have two aspects/tenses (perfect and imperfect), five moods (indicative, subjunctive, jussive, imperative and energetic), and two voices (active and passive) for transitive verbs.

Verb inflection occurs through the concatenation of prefixes and suffixes with the stem. Infix modifications (stem changes) may occur in the presence of certain syntactic features and "weak" consonants ('w' or 'y') as radicals, and through a change of stem vowel between the

perfect and imperfect. In verbs of pattern 1 (the simplest triliteral pattern), this is not generally predictable. Several of the patterns share the same prefixes and/or suffixes, but these interact with stems beginning and ending in weak and other consonants to yield a complexly varied orthography in inflected forms.

There are 13 person-number-gender combinations, as Arabic only distinguishes between singular and plural number for 1st person, but also adds dual and gender distinctions for 2nd and 3rd persons.

3.1.2. Arabic Noun and Adjective Morphology

Arabic nouns and adjectives are also formed via the intersection of three or four consonant radicals with vowel and consonant patterns. They present an inflectional paradigm based on definiteness (definite, indefinite), case (nominative, genitive, accusative) and number (singular, dual, plural).

A minority of nouns have *sound* plurals, which are regularly formed through the addition of suffixes that depend on grammatical gender and case. More common are *broken* plurals, which entail stem changes according to various patterns that are not predictable from the singular noun pattern. Broken plurals behave, under inflection, similarly to singular nouns. Dual inflection occurs through regular prefixes.

Adjectives are not strongly distinguished from nouns, and also may have sound and broken plurals. However, due to the special agreement rules of Arabic, adjectives modifying plural non-human nouns take a singular feminine inflection, so that relatively few adjectives are actually ever pluralized.

3.2. Initial Extensions to MORPHĒ

We began by using MORPHĒ to describe Arabic verbal morphology. The richness of the inflectional paradigm for verbs and the stem changes occurring in verbs with weak radicals, combined with the restriction that transformational rules could only be attached to leaf nodes of the MFH, and only one rule per node, gave rise to extensive and highly redundant descriptions for even a subset of the full range of inflected forms. In an effort to alleviate these problems, we split the generation process into two steps: 1) generation of stems and 2) generation of prefixes and suffixes. The MFH was given separate subtrees for these two processes and MORPHĒ was called twice to obtain the final inflected forms (Cavalli-Sforza et al., 2000). Our representation of the base form of lexical items was stem-based, since it is stems and not roots that have specific meaning. Stem changes, as well as prefix and suffix additions, were effected through transformational rules but, in order to decide which vowel change to apply to a stem in the imperfective, we relied on information associated with an item in the lexicon.

Some theoretical and computational treatments of morphology have given equal status to morphemes representing meanings (lexemes) and to grammatical morphemes such as gender and number inflections, placing them in separate dictionaries (e.g. Kiraz, 2000; Buckwalter, 2004). MORPHĒ's treatment of stems as primary morphemes and of inflectional affixes as transformations applied to stems deviates from that conception of morphology but agrees well with the theory of Lexeme-Based Morphology (Aronoff, 1994; Beard, 1995; Soudi et al., 2001). The next step was to consider inflection of nouns and, in particular, how to handle broken plurals within the same framework (Soudi et al., 2002). Although the singular and plural have a different stem, they are forms of the same *lexeme* and should not be treated as separate lexicon entries. However, since the plural stem cannot be predictably derived from the singular, it must be associated with the singular in the lexicon as an alternate base form.

Experience with the original MORPHĒ and comparison with Lexeme-Based Morphology, suggested a number of theoretically motivated extensions to MORPHĒ's description language:

1) *Allomorph* declarations support stem alternations linked to specific lexical items, are associated with a node in the MFH, and direct MORPHĒ to look for an alternate base form in the input FS.
2) *Default rules*, attached to pre-leaf nodes, describe transformations associated with most but not all nodes in an MFH subtree. The more specific (exceptional) transformations are attached to the leaf nodes.
3) *Equivalence declarations* express the equivalence of transformations across different portions of the MFH. This construct clearly corresponds to the *rules of referral* postulated by Lexeme-Based Morphology.
4) *Implicit equivalencing* of nodes states that the same node can be reached through different feature-value paths in the MFH, turning the hierarchy into a graph.

The above language extensions allowed us to describe the fully diacritized inflected forms of strong and hollow verbs (verbs with a weak middle radical) and sound and broken noun plurals for MSA even more compactly, showing explicitly which forms behave similarly and reducing the redundancy of the description.

Additional system extensions were targeted at facilitating the development of large morphology descriptions. The Enhanced MORPHĒ tool (EMORPHĒ) (Cavalli-Sforza & Soudi, 2003) accepts morphology descriptions spread across multiple files. It also includes a facility for thoroughly testing the morphology description by automatically generating all forms of a collection of representative lexical items and comparing them to expected results.

3.3. EMORPHĒ's Limitations

While EMORPHĒ significantly enhances original MORPHĒ, it still falls short of providing an optimally concise and elegant framework for describing the morphology of languages such as MSA. We characterize those languages as displaying a richness of inflectional forms distinguished by classes of stem alternations, prefixes and suffixes that are shared by some but not all forms. Our goal was to more concisely capture regularities in the morphological descriptions of such languages while also improving runtime efficiency.

Conciseness: Because in (E)MORPHĒ a single rule can be attached to a node, a rule must take care of all necessary transformations, which makes rules monolithic and redundant across forms that share similar affix changes. Recall that this problem was the motivation for initially splitting the generation process into two phases; it was further alleviated, but not completely eliminated, by equivalence declarations in EMORPHĒ.

Efficiency: In the two-stage approach still used in EMORPHĒ to generate the fully inflected form of a lexical item, redundant work is performed in checking the features in the input FS twice, once for determining the appropriate stem, then again for determining the prefix and suffix. Since the stem subtree is relatively shallow, at least for nouns and for sound and hollow verbs, the cost in time is not high, but avoiding it altogether would be better. Furthermore, the two-stage process complicates extending EMORPHĒ to perform analysis.

The next section describes how our current work seeks to address the two issues of conciseness and efficiency.

4. IMORPHĒ

IMORPHĒ, is an inheritance-and-equivalence based reformulation of MORPHĒ. It retains MORPHĒ's original conception of morphology description based on an MFH and transformational rules, and the equivalence and allomorph declarations of EMORPHĒ, but casts them in an inheritance-based framework with more modular rules. Transformational rules and allomorph declarations can be attached to any node in the hierarchy and inherited, canceled, or overridden by children nodes, thereby generalizing EMORPHĒ's default rules. The restriction of one rule per node is removed: multiple rules can be attached to any node or defined as global rules to be used by interested nodes. Via an extended equivalence declaration, a node can "borrow" (use) some or all rules present on other nodes (or even just some of their clauses), and/or use a global rule. Finally, a node can change the order of application of its inherited and/or borrowed rules.

The above extensions, described below in greater detail, required significant additions to the morphology description language, and the morphology description compiler must do significantly more work to collect and adjust the information associated with each inflected form. However, runtime generation of inflected forms is performed in a single pass through the MFH, as in IMORPHĒ's predecessors. In addition IMORPHĒ retains backwards compatibility by being able to compile morphology descriptions that use only MORPHĒ and EMORPHĒ declarations.

4.1. The Morphology Description Language

A full Extended BNF specification of IMORPHĒ's morphology description language is beyond the scope of this paper, but we highlight below its major constructs and point out its relationship with (E)MORPHĒ's constructs.

4.1.1. MORPH-FORM Declarations

IMORPHĒ's MFH is built through a set of **morph-form** declarations. The simplest syntax is:

```
(MORPH-FORM <node> <parent> <FS-pieces>)
```

where <node> is the new node being created, <parent> is its parent node, and <FS-pieces> is one or more feature-value pairs that distinguish the node from its parent and, if present in a lexical item's FS at runtime, will allow it to reach this node.

This simple declaration, already present in (E)MORPHĒ, is supplemented in IMORPHĒ by three optional slots (**:allomorph**, **:rules**, **:inherit**) whose contents allow associating with the node, locally and more precisely, allomorph, rule and inheritance information.

The extended syntax of a morph-form declaration, which gives IMORPHĒ an object-oriented flavor, is then:

```
(MORPH-FORM <node> <parent> <FS-pieces>
    :allomorph <allomorph-spec>
    :rules     <rules-spec>
    :inherit   <inheritance-spec>
    )
```

The **:allomorph** and **:rules** slots are described further below. The **:inherit** slot is used by an implicitly equivalenced node (a node with the same name as another node but reached through a different FS path through the MFH), in order to specify through which path to inherit information. Additional **:position** subslots in each of the **:allomorph** and **:rules** slot are used to reorder allomorph substitutions and rules acquired through inheritance.

4.1.2. MORPH-ALLOMORPH Declarations

The **morph-allomorph** declaration, introduced in EMORPHĒ, associates with a node a directive to search in the FS for an alternate base form to which apply further transformations. Only one allomorph is allowed on any given node. The syntax used in EMORPHĒ is:

```
(MORPH-ALLOMORPH <node> <feature>)
```

where <feature> is the name of the FS feature whose value contains the alternate base form. In IMORPHĒ, this syntax is retained and extended with an optional <position> specifier, which allows changing the order of application of this operation relative to other rules associated with a node. A **morph-allomorph** declaration can be placed anywhere in the file after the <node> itself has been created through a **morph-form** declaration. This mechanism for declaring allomorph information, detached from node creation, is retained for backwards compatibility. In IMORPHĒ, the same effect is obtained by attaching the information directly to the node itself through the **:allomorph** slot, in an **:own** subslot. The **:allomorph** slot further permits specifying whether the node cancels allomorph information coming from an ancestor or overrides it with information locally specified on the node or borrowed through an equivalence (see below) to another node.

4.1.3. MORPH-(G)RULE Declarations

A **morph-rule** declaration attaches a transformational rule to a node. The basic syntax of a rule is:

```
(MORPH-RULE <node> <clause>*)
```

A <clause> is defined as:

```
(<regexp-pattern> <operation>*)
```

where <regexp-pattern> is a regular expression string that matches against the base form of a lexical item and <operation> is a list of operations to be applied to that base form if the test is successful. This rule declaration syntax is retained in IMORPHĒ but augmented and supplemented in several ways.

Firstly, in IMORPHĒ, unlike its predecessors, more than one rule can be attached to a node. If there are multiple **morph-rule** declarations naming a node, they will be attached to the node in the order in which they are read in. However, as for **morph-allomorph** declarations, the above rule declaration syntax is augmented by allowing as the last element a <position> specifier which

states the order in which the rule should be applied to the base form relative to other rules and allomorph substitution:

```
(MORPH-RULE <node> <clause>* [<position>])
```

Secondly, in order to make rules less monolithic, that is, to allow pieces of rules to be borrowed and used by other nodes, both clauses and operations are optionally named (and should be named if they are to be borrowed).

Thirdly, a new **morph-grule** declaration allows a "global" rule (a rule that is not a priori associated with any node) to be defined, with the expectation that it will be used by multiple other nodes.

Finally, rules can be attached directly to a node in a **morph-form** declaration through the **:rules** slot and **:own** subslot, where they can also be optionally be named (if they are to be borrowed). In addition to declaring rules for a node, the **:rules** slot specifies whether the node borrows rules and rule pieces from other nodes, uses global rules, cancels rule information coming from an ancestor and/or supplements it with rule pieces from other nodes and/or a global rule. Rules attached to a node via the **:rules** slot and/or via a **morph-rule** declaration, can be reordered with respect to each other and an allomorph substitution. It is the compiler's job to cumulate and reorder the rule information.

4.1.4. MORPH-EQUIVALENCE Declarations

EMORPHĒ uses a **morph-equivalence** declaration, to state that one node was to be considered equivalent to another node and could borrow rules associated with that node. A node can be equivalenced to only one other node but could have multiple nodes equivalenced to it. The syntax, retained in IMORPHĒ, is:

```
(MORPH-EQUIVALENCE <common-node>
                   ( <equiv-node>+ ) )
```

and says that each of the <equiv-node>s is like <common-node>. The latter could be an actual node in the MFH, or a virtual node to which rule information could be attached through **morph-rule** declarations.

IMORPHĒ also allows information to be borrowed from other nodes more selectively through the **:allomorph** and **:rules** slots in the **morph-form** declaration.

4.2. The IMORPHĒ Compiler

In (E)MORPHĒ, rules and allomorphs are attached only to pre-leaf and leaf nodes of the MFH and compilation of a morphology description required a single pass through the MFH in order to generate the CommonLisp functions that then generate morphological forms at runtime. However, in order to achieve a minimally redundant morphological description for a language such as Arabic, where morphological forms share several stem, prefix and suffix transformations, the MFH was split into two parts (stem/infix and prefix plus suffix transformations) and the generation process at runtime required two passes. IMORPHĒ replaces the simple compilation process with one that requires several passes, but, once the morphology description is fully compiled, runtime generation occurs in a single step.

The first pass reads in all morphology description files, checking the syntax of declarations and building the MFH. Allomorph, rule, equivalence and inheritance information is attached to the MFH nodes in a preliminary way that closely mirrors the structure of the description files.

In the second pass, the compiler navigates the MFH using the preliminary attachment information to actually find and attach to the nodes the appropriate allomorphs and rules. Inherited information is percolated down, canceled or overridden. Information from equivalences is retrieved, selected or overridden. Finally, the results are assembled, resulting in the possible creation of new rules from pieces of existing rules.

The third pass applies ordering constraints on allomorphs and rules to produce a final sequence of operations (allomorph substitutions and transformations) for each node in the MFH.

The fourth and final pass reuses parts of the old MORPHĒ compiler to create the Common Lisp functions that implement the compiled morphology description.

Compilation may stop at the end of the first or the second pass if syntactic or semantic errors are found or names cannot be resolved. The third pass may produce warnings about ambiguous reordering, but compilation will otherwise run all the way through.

4.3. Using IMORPHĒ with Arabic

At the time of this writing, the IMORPHĒ compiler is fully designed but still under implementation. We are testing each compilation phase using morphology descriptions developed while using EMORPHĒ for strong and hollow (weak middle radical) Arabic verbs and noun inflection, and augmenting the test suite with some defective (weak final radical) and assimilated (weak initial radical) verbs.

Allomorph attributions and other stem change information are attached higher in the MFH than prefix/suffix transformational rules. In generating the inflected forms, the resulting stem modifications are performed before prefix and suffix additions. Inheritance allows prefix and suffix rules to be attached at the point of highest coverage and overridden in specific cases. Equivalencing, multiple rules, and rule naming allow rules to be attached to a single appropriate node, and borrowed by other nodes in the MFH. The result is a morphology description where rules are specified only once, but may be used by multiple nodes in the MFH, even distant ones. The resulting morphology description is relatively simple, minimally redundant, and shows clearly the similarities between different forms, even those that are distant from each other in the MFH.

5. Conclusions and Future Work

We have described the evolution, design and current state of the IMORPHĒ morphology description compiler. While IMORPHĒ's morphology description language incorporates several ideas that conveniently address characteristics of Modern Standard Arabic morphology, IMORPHĒ is language independent. Its use with MSA has pushed the tool's development in a direction that supports minimally redundant morphological descriptions and makes explicit the similarities of related morphological forms.

At present, work on the compiler is still in progress and, when completed, will give IMORPHĒ full generation capability. Turning IMORPHĒ into an analyzer will be one of the next steps but one further extension, prior to

proceeding to work on analysis, will be the addition of optional rules. These will allow IMORPHĒ to support prefixes – such as prepositions and conjunctions – and suffixes – such as possessive and direct object pronouns – that are attached to words but are not part of the inflectional morphology of the language and may or may not be present. We expect the final tool, including the testing system, to be available through Carnegie Mellon University's Language Technologies Institute under a no cost licensing agreement for research use.

6. References

Abuleil, S., Alsamara, K. & Evens, M. (2002). Acquisition System for Arabic Noun Morphology. In *Proceedings of the Workshop on Computational Approaches to Semitic Languages*, Conference of the Association for Computational Linguistics (ACL 2002), Philadelphia, pp. 19-26.

Aronoff, M. (1994). *Morphology by Itself: Stems and Inflectional Classes*. Cambridge, MA: MIT Press.

Beard, R. (1995). *Lexeme-Morpheme Base Morphology: A General Theory of Inflection and Word Formation*. New York, NY: State University of New York Press.

Beesley, K., Buckwalter, T. & Newton, S. (1989). Two-Level Finite-State Analysis of Arabic Morphology. In *Proceedings of the Seminar on Bilingual Computing in Arabic and English*, Cambridge, England.

Beesley, K. (1996). Arabic Finite-State Morphological Analysis and Generation. In *Proceedings COLING'96*, Vol. 1, pp. 89-94.

Buckwalter, T. (2004). *Buckwalter Arabic Morphological Analyzer Version 2.0*. Linguistic Data Consortium, University of Pennsylvania. LDC Catalog Number: LDC2004L02, www.ldc.upenn.edu/Catalog/.

Cavalli-Sforza, V. & Soudi A. (forthcoming). Arabic Computational Morphology: A Tradeoff between Multiple-Operations and Multiple-Stems. In *Arabic Computational Morphology: Knowledge-based and Empirical Methods*. In A. Soudi, A. Van den Bosch, G. Neumann (Eds.). Arabic *Computational Morphology: Knowledge-based and Empirical Methods*, Kluwer/Springer's series on Text, Speech and Language Technology.

Cavalli-Sforza, V. & Soudi A. (2003), Enhancements to a Morphological Generator to Capture Arabic Morphology. In *Proceedings of the Eighth International Symposium on Social Communication*, Center of Applied Linguistics, Santiago de Cuba, pp. 565-570.

Cavalli-Sforza, V., Soudi, A., & Mitamura, T. (2000), Arabic Morphology Generation Using a Concatenative Strategy. In *Proceedings of the First Meeting of the North American Chapter of the Association for Computational Linguistics* (NAACL 2000), Seattle, pp. 86-93.

Darwish, K. (2002). Building a Shallow Arabic Morphological Analyzer in One Day. In *Proceedings of the Workshop on Computational Approaches to Semitic Languages*. Conference of the Association for Computational Linguistics (ACL 2002), Philadelphia, pp. 9-18.

Dichy, J. & Farghaly, A. (2003). Roots & Patterns vs. Stems plus Grammar-Lexis Specifications: on what basis should a multilingual lexical database centred on Arabic be built?. In *Proceedings of the Workshop on Machine Translation for Semitic Languages: Issues and Approaches*, Ninth Machine Translation Summit, New Orleans, pp. 1-8.

Finkel, R. & Stump, G. (2002). Generating Hebrew verb morphology by default inheritance hierarchies. In *Proceedings of the Workshop on Computational Approaches to Semitic Languages*. Conference of the Association for Computational Linguistics (ACL 2002), Philadelphia, pp. 9-18.

Finkler, W. & Neumann, G. (1988). MORPHIX. A Fast Realization of a Classification-Based Approach to Morphology. In H. Trost (Ed.) 4. *Österreichische Artificial-Intelligence-Tagung. Wiener Workshop - Wissensbasierte Sprachverarbeitung. Proceedings*. Berlin etc.: Springer, pp. 11-19.

Habash, Nizar (2004). Large Scale Lexeme Based Arabic Morphological Generation. In *Proceedings of Traitement Automatique du Langage Naturel* (TALN-04). Fez, Morocco.

Itai, A. & Segal, E. (2003). A Corpus Based Morphological Analyzer for Unvocalized Modern Hebrew In *Proceedings of the Workshop on Machine Translation for Semitic Languages: Issues and Approaches*, Ninth Machine Translation Summit, New Orleans, pp. 29-36.

Kiraz, G. (1998). Arabic Computational Morphology in the West. In *Proceedings of the Sixth International Conference and Exhibition on Multi-lingual Computing*, Cambridge.

Kiraz, G. (2000). A Multi-tiered Nonlinear Morphology using Multi-tape Finite State Automata: A Case Study on Syriac and Arabic. In *Computational Linguistics*, 26 (1), pp. 77-105.

Koskenniemi, K. (1983). *Two-level morphology: A General Computational Model for Word-Form Recognition and Production*. Ph.D. Thesis, University of Helsinki.

Leavitt, J.R. (1994). *MORPHĒ: A Morphological Rule Compiler*. Technical Report: CMU-CMT-94-MEMO.

Macks, A. (2002). Parsing Akkadian Verbs with Prolog. In *Proceedings of the Workshop on Computational Approaches to Semitic Languages*. Conference of the Association for Computational Linguistics (ACL 2002), Philadelphia, pp. 3-8.

Marsi, E., Van den Bosch, A., & Soudi, A. (2005). Memory-based morphological analisis generation and part-of-speech tagging of Arabic. In *Proceedings of the Workshop on Computational Approaches to Semitic Languages*, Conference of the Association for Computational Linguistics, Ann Arbor, pp. 1-8

Nyberg, E., & Mitamura, T. (1992). The KANT system: Fast, accurate, high-quality translation in practical domains. In *Proceedings of COLING-9*.

Soudi, A., Cavalli-Sforza, V., & Jamari, A. (2002). The Arabic Noun System Generation. In *Proceedings of the International Symposium on The Processing of Arabic*, University of Manouba, Tunisia, pp. 69-87.

Soudi, A., Cavalli-Sforza, V., & Jamari, A. (2001). A Computational Lexeme-Based Treatment of Arabic Morphology. In *Proceedings of the Arabic Natural Language Processing Workshop*, Conference of the Association for Computational Linguistics (ACL 2001), Toulouse, pp. 155-162.