PBML

The Prague Bulletin of Mathematical Linguistics
NUMBER 88 DECEMBER 2007 5–30

Functional Arabic Morphology

Dissertation Summary

Otakar Smrž

Abstract

This is a summary of the author's PhD dissertation defended on September 17, 2007 at the Faculty of Mathematics and Physics, Charles University in Prague. The results comprised in the thesis were obtained within the author's doctoral studies in Mathematical Linguistics during the years 2001–2007. The complete dissertation is available via `http://sourceforge.net/projects/elixir-fm/`.

1. Introduction

Functional Arabic Morphology is a formulation of the Arabic inflectional system seeking the working interface between morphology and syntax. ElixirFM is its high-level implementation that reuses and extends the Functional Morphology library for Haskell (Forsberg and Ranta, 2004), yet the treatment of the language-specific issues constitutes our original work.

In the thesis (Smrž, 2007), we develop a computational model of the morphological processes in Arabic. With this system, we are able to derive and inflect words, as well as to analyze the structure of word forms and to recognize their grammatical functions.

The approach to building our morphological model strives to be comprehensive with respect to linguistic generalization, and high-level and modern with respect to the programming techniques that we employ. We describe the linguistic concept and try to implement it in a very similar, yet abstract way, using the declarative functional programming language Haskell. We emphasize the flexibility of our system, its reusability and extensibility.

1.1. Morphological Models

One can observe several different streams both in the computational and the purely linguistic modeling of morphology. Some are motivated by the need to analyze word forms as to their compositional structure, others consider word inflection as being driven by the underlying system of the language and the formal requirements of its grammar.

There are substantial discrepancies between the grammatical descriptions of Arabic represented e.g. by (Fischer, 2001) or (Holes, 2004), and the information that the available morphological computational systems provide. One of the reasons is that there is never a complete consensus on what the grammatical description should be. The other source of the incompatibility lies in the observation that many implementations overlook the principal difference between the function and the form of a linguistic symbol.

Many of the computational models of Arabic morphology, including in particular (Beesley, 2001), (Ramsay and Mansur, 2001) or (Buckwalter, 2002), are *lexical* in nature, i.e. they tend to treat inflectional affixes just like full-fledged lexical words. As they are not designed in connection with any syntax–morphology interface, their interpretations are destined to be *incremental*. That means that the only clue for discovering the morphosyntactic properties of a word is through the explicit affixes and their prototypical functions.

Some signs of a *lexical–realizational* system can be found in (Habash, 2004). The author mentions and fixes the problem of underdetermination of inherent number with plurals, when developing a generative counterpart to (Buckwalter, 2002).

The computational models in (Cavalli-Sforza, Soudi, and Mitamura, 2000) and (Habash, Rambow, and Kiraz, 2005) attempt at the *inferential–realizational* direction. Unfortunately, they implement only sections of the Arabic morphological system. The Arabic resource grammar in the Grammatical Framework (El Dada and Ranta, 2006) is perhaps the most complete inferential–realizational implementation to date. Its style is compatible with the linguistic description in e.g. (Fischer, 2001) or (Badawi, Carter, and Gully, 2004), but the lexicon is now very limited and some other extensions for data-oriented computational applications are still needed.

ElixirFM, the implementation of the system developed in this thesis, is inspired by the methodology in (Forsberg and Ranta, 2004) and by functional programming, just like the Arabic GF is (El Dada and Ranta, 2006). Nonetheless, ElixirFM reuses the Buckwalter lexicon (Buckwalter, 2002) and the annotations in the Prague Arabic Dependency Treebank (Hajič et al., 2004b), and implements a yet more refined linguistic model.

In our view, influenced by the Prague linguistic school and the theory of Functional Generative Description (Sgall, 1967, Sgall, Hajičová, and Panevová, 1986, Panevová, 1980, Hajičová and Sgall, 2003), the task of morphology should be to analyze word forms of a language not only by finding their internal structure, i.e. recognizing morphs, but even by *strictly* discriminating their functions, i.e. providing the true morphemes. Conceived in such a way, it should be *completely* sufficient to generate the word form that represents a lexical unit and features all grammatical categories (and structural components) required by context, purely from the information comprised in the analyses.

It appears from the literature on most other implementations (many summarized in (Al-Sughaiyer and Al-Kharashi, 2004)) that the Arabic computational morphology has understood its role in the sense of operations with morphs rather than morphemes (cf. (El-Sadany and Hashish, 1989)), and has not concerned itself systematically and to the necessary extent with the role of morphology for syntax. In other terms, the syntax–morphology interface has not been clearly established and respected.

The outline of formal grammar in (Ditters, 2001), for instance, works with grammatical categories like number, gender, humanness, definiteness, but one cannot see which of the existing systems could provide for this information correctly, as they misinterpret some morphs for bearing a category, and underdetermine lexical morphemes in general as to their intrinsic morphological functions. Nowadays, the only exception is the Arabic Grammatical Framework (El Dada and Ranta, 2006, Dada, 2007), which implements its own morphological and syntactic model.

Certain syntactic parsers, like (Othman, Shaalan, and Rafea, 2003), may resort to their own morphological analyzers, but still, they do not get rid of the form of an expression and only incidentally introduce truly functional categories. In syntactic considerations they often call for discriminative extra-linguistic features instead. Commercial systems, e.g. (Chalabi, 2004), do not seem to overcome this interference either.

1.2. Reused Software

The ElixirFM implementation of Functional Arabic Morphology would not have come into existence were it not for many open-source software projects that we could use during our work, or by which we got inspired.

ElixirFM and its lexicons are licensed under GNU GPL and are available on `http://sourceforge.net/projects/elixir-fm/`, along with the other accompanying software (MorphoTrees, Encode Arabic) and the source code of the thesis (ArabTEX extensions, TreeX).

ElixirFM 1.0 is intended for use with the Hugs interactive interpreter of Haskell, available for a number of platforms via `http://haskell.org/hugs/`.

Buckwalter Arabic Morphological Analyzer The bulk of lexical entries in ElixirFM is extracted from the data in the Buckwalter lexicon (Buckwalter, 2002). We devised an algorithm in Perl using the morphophonemic patterns of ElixirFM that finds the roots and templates of the lexical items, as they are available only partially in the original, and produces the ElixirFM lexicon in customizable formats for Haskell and for Perl.

Functional Morphology Library Functional Morphology (Forsberg and Ranta, 2004) is both a methodology for modeling morphology in a paradigmatic manner, and a library of purposely language-independent but customizable modules and functions for Haskell. It partly builds on the Zen computational toolkit for Sanskrit (Huet, 2002). Functional Morphology is also related to the Grammatical Framework, cf. (El Dada and Ranta, 2006) and `http://www.cs.chalmers.se/~markus/FM/`.

TrEd Tree Editor TrEd `http://ufal.mff.cuni.cz/~pajas/tred/` is a general-purpose graphical editor for trees and tree-like graphs written by Petr Pajas. It is implemented in Perl and is designed to enable powerful customization and macro programming. We have extended TrEd with the annotation mode for MorphoTrees.

1.3. Original Contributions

The following are the original contributions and proposals of the present study:

(i) Recognition of functional versus illusory morphological categories, definition of a minimal but complete system of inflectional parameters in Arabic
(ii) Morphophonemic patterns and their significance for the simplification of the model of morphological alternations
(iii) Inflectional invariant and its consequence for the efficiency of morphological recognition in Arabic
(iv) Intuitive notation for the structural components of words
(v) Conversion of the Buckwalter lexicon into a functional format resembling printed dictionaries
(vi) ElixirFM as a general-purpose model of morphological inflection and derivation in Arabic, implemented with high-level declarative programming
(vii) Abstraction from one particular orthography affecting the clarity of the model and extending its applicability to other written representations of the language
(viii) MorphoTrees as a hierarchization of the process of morphological disambiguation
(ix) Expandable morphological positional tags, restrictions on features, their inheritance
(x) Open-source implementations of ElixirFM, Encode Arabic, MorphoTrees, and extensions for ArabTEX

2. Writing & Reading Arabic

In the context of linguistics, morphology is the study of word forms. In formal language theory, the symbols for representing words are an inseparable part of the definition of the language. In natural languages, the concept is a little different—an utterance can have multiple representations, depending on the means of communication and the conventions for recording it. An abstract computational morphological model should not be limited to texts written in one customary orthography.

This chapter explores the interplay between the genuine writing system and the transcriptions of Arabic. We introduce in detail the ArabTEX notation, a morphophonemic transliteration scheme adopted as the representation of choice for our general-purpose morphological model. We then discuss the problem of recognizing the internal structure of words given the various possible types of their record.

2.1. ArabTEX Notation

The ArabTEX typesetting system (Lagally, 2004) defines its own Arabic script meta-encoding that covers both contemporary and historical orthography. The notation is human-readable and very natural to write with. Its design is inspired by the standard phonetic transcription of Arabic, which it mimics, yet some distinctions are introduced to make the conversion to the original script or the transcription unambiguous.

Unlike other transliteration concepts based on the strict one-to-one substitution of graphemes, ArabTEX interprets the input characters in context in order to get their proper meaning. Deciding the glyphs of letters (initial, medial, final, isolated) and their ligatures is not the issue of encoding, but of visualizing of the script. Nonetheless, definite article assimilation, inference of *hamza* carriers and silent *ʾalif*s, treatment of auxiliary vowels, optional quoting of diacritics or capitalization, resolution of notational variants, and mode-dependent processing remain the challenges for parsing the notation successfully.

ArabTEX's implementation is documented in (Lagally, 1992), but the parsing algorithm for the notation has not been published except in the form of the source code. The TEX code is organized into deterministic-parsing macros, yet the complexity of the whole system makes consistent modifications or extensions by other users quite difficult.

We describe our own implementations of the interpreter in Chapter 9, where we show how to decode the notation and its proposed extensions. To encode the Arabic script or its phonetic transcription into the ArabTEX notation requires heuristic methods, if we want to achieve linguistically appropriate results.

2.2. Recognition Issues

Arabic is a language of rich morphology, both derivational and inflectional. Due to the fact that the Arabic script usually does not encode short vowels and omits some other important phonological distinctions, the degree of morphological ambiguity is very high.

Besides this complexity, Arabic orthography prescribes to concatenate certain word forms with the preceding or the following ones, possibly changing their spelling and not just leaving out the whitespace in between them. This convention makes the boundaries of lexical or syntactic units, which need to be retrieved as tokens for any deeper linguistic processing, obscure, for they may combine into one compact string of letters and be no more the distinct 'words'.

Thus, the problem of disambiguation of Arabic encompasses not only diacritization (discussed in (Nelken and Shieber, 2005)), but even tokenization, lemmatization, restoration of the structural components of words, and the discovery of their actual morphosyntactic properties, i.e. morphological tagging (cf. (Hajič et al., 2005), plus references therein). These subproblems, of course, can come in many variants, and are partially coupled.

3. Morphological Theory

This chapter defines lexical words as the tokens on which morphological inflection proper will operate. We explore what morphosyntactic properties should be included in the functional model. We discuss the linguistic and computational views on inflectional morphology. Further, we are concerned with Arabic morphology from the structural perspective, designing original morphophonemic patterns and presenting roots as convenient inflectional invariants.

3.1. Functional and Illusory Categories

Functional Arabic Morphology endorses the inferential–realizational principles in the morphological theory (cf. (Stump, 2001)). It re-establishes the system of inflectional and inherent morphosyntactic properties (or grammatical categories or features, in the alternative naming) and discriminates precisely the senses of their use in the grammar. It also deals with syncretism of forms (cf. (Baerman, Brown, and Corbett, 2006)) that seems to prevent the resolution of the underlying categories in some morphological analyzers.

In the thesis, we offer examples of morphological analyses disclosing that grammatical descriptions cannot dispense with a single category for number or for gender, but rather, that we should always specify the sense in which we mean these:

functional category is introduced as the morphosyntactic property that is involved in grammatical considerations; we further divide functional categories into

- **logical categories** on which agreement with numerals and quantifiers is based
- **formal categories** controlling other kinds of agreement or pronominal reference

illusory category denotes the value derived merely from the morphs of an expression

Does the classification of the senses of categories actually bring new quality to the linguistic description? We explore in detail the extent of the differences in the values assigned. It may, of course, happen that the values for a given category coincide in all the senses. However, promoting the illusory values to the functional ones is in principle conflicting:

(i) Illusory categories are set only by a presence of some 'characteristic' morph, irrespective of the functional categories of the whole expression.

(ii) If no morph 'characteristic' of a value surrounds the word stem and the stem's morpheme does not have the right information in the lexicon, then the illusory category remains unset. It is the particular issue with the internal/broken plural in Arabic, for which the illusory analyses do not reveal any values of number or gender.

The problem concerns every nominal expression individually and pertains to some verbal forms, too. Functional Arabic Morphology makes the functional gender and number information available thanks to the lexicon that can stipulate some properties as inherent to some lexemes, and thanks to the paradigm-driven generation that associates the inflected forms with the desired functions directly.

3.2. The Pattern Alchemy

In Functional Arabic Morphology, patterns constitute the inventory of phonological melodies of words, regardless of the other functions of the words. Morphophonemic patterns abstract from the consonantal root, which is often recognized or postulated on etymological grounds. Other types of patterns, like the decomposition into separate CV patterns and vocalisms, can be derived from the morphophonemic patterns.

Fischer (2001) uses patterns that abstract away from the root, but can include even inflectional affixes or occasionally restore weak root consonants. For instance, we can find references to patterns like *ʾafʿala* for *ʾaḥsana* أَحْسَنَ ‘he did right’ or *ʾahdā* أَهْدَى ‘he gave’, but *ʾafʿalu* for *ʾaʿlā* أَعْلَى ‘higher’. In our model, the morphophonemic pattern pertains to the morphological stem and reflects its phonological qualities. Thus, our patterns become `HaFCaL` for *ʾaḥsana* أَحْسَنَ, while `HaFCY` for both *ʾahdā* أَهْدَى and *ʾaʿlā* أَعْلَى.

Beesley (1998) uses the term ‘morphophonemic’ as ‘underlying’, denoting the patterns like `CuCiC` or `staCCaC` or `maCCuuC`. Yet, he also uses the term for anything but the surface form, cf. “an interdigitated but still morphophonemic stem” or “there may be many phonological or orthographical variations between these morphophonemic strings and their ultimate surface pronunciation or spelling” (Beesley, 1998).

Kay (1987) gives an account of finite-state modeling of the nonconcatenative morphological operations. He calls CV patterns ‘prosodic templates’, both terms following (McCarthy, 1981). For further terminological explanations, cf. ((Kiraz, 2001), pages 27–46).

We build on morphophonemic patterns rather than on CV patterns and vocalisms. Words like *istaǧāb* اِسْتَجَاب ‘to respond’ and *istaǧwab* اِسْتَجْوَب ‘to interrogate’ have the same underlying VstVCCVC pattern, thus the information on CV patterns alone would not be enough to reconstruct the differences in the surface forms. Morphophonemic patterns, in this case `IstaFAL` and `IstaFCaL`, can easily be mapped to the hypothetical CV patterns and vocalisms, or linked with each other according to their relationship. Morphophonemic patterns deliver more information in a more compact way.

With this approach, we also get a more precise control over the actual word forms—we explicitly confirm that the ‘word’ the pattern should create does undergo the implied transformations. One can therefore speak of ‘weak patterns’ rather than of ‘weak roots’.

The idea of pre-computing the phonological constraints within CV patterns into the ‘morphophonemic’ ones is present in (Yaghi and Yagi, 2004), but is applied to verbs only and is perhaps not understood in the sense of a primary or full-fledged representation ((Yaghi and Yagi, 2004), sec. 5):

> The transformation may be made on the morphological pattern itself, thus producing a sound surface form template. … A coding scheme is adopted that continues to retain letter origins and radical positions in the template so that this will not affect [the author’s model of] affixation. … The surface form template can be rewritten as إِh_F2تَّh_M0َh_L2ى `AiF2t~aM0aL2Y`. This can be used to form stems such as إِتَّدَى `Ait~adaY` by slotting the root ودي `wdy`.

Yaghi’s templates are not void of root-consonant ‘placeholders’ that actually change under inflection, cf. h_F2 h_L2 indexed by the auxiliary integers to denote their ‘substitutability’. The template, on the other hand, reflects some of the orthographic details and includes Form VIII assimilations that can be abstracted from, cf. esp. the تَّ `t~a` group.

With Functional Arabic Morphology, the morphophonemic pattern of *ittadā* إِتَّدَى is simply `IFtaCY`, the root being *wdy* ودي. One of its inflected forms is `IFtaCY |<< "tumA"` *ittadaytumā* إِتَّدَيْتُمَا ‘the two of you accepted compensation’, to follow again the example in (Yaghi and Yagi, 2004). We describe the essence of this notation in Chapter 5.

CV templates are viewed from the perspective of moraic templates in the Prosodic Morphology (McCarthy and Prince, 1990), later discussed by (Kiraz, 2001) within his development of a multitier nonlinear morphological model. Given that we can define a mapping from morphophonemic templates into prosodic or moraic templates, which we easily can, we claim that the prosodic study of the templates is separable from the modeling of morphology.

3.3. The Inflectional Invariant

In our approach, we define roots as sequences of consonants. In most cases, roots are triliteral, such as *k t b* كتب, *q w m* قوم, *d s s* دسس, *r ʾ y* رأي, or quadriliteral, like *d ḥ r ǧ* دحرج, *ṭ m ʾ n* طمأن, *z l z l* زلزل.

Roots in Arabic are, somewhat by definition, inflectional invariants. Unless a root consonant is weak, i.e. one of *y*, *w* or *ʾ*, and unless it assimilates inside a Form VIII pattern, then this consonant will be part of the inflected word form. This becomes apparent when we consider the repertoire and the nature of morphophonemic patterns.

The corollary is that we can effectively exploit the invariant during recognition of word forms. We can check the derivations and inflections of the identified or hypothesized roots only, and need not inflect the whole lexicon before analyzing the given inflected forms in question.

While this seems the obvious way in which learners of Arabic analyze unknown words to look them up in the dictionary, it contrasts strongly with the practice in the design of computational analyzers, where finite-state transducers (Beesley and Karttunen, 2003), or analogously tries (Forsberg and Ranta, 2004, Huet, 2002), are most often used. Of course, other languages than Arabic need not have such convenient invariants.

4. Impressive Haskell

Haskell is a purely functional programming language based on typed λ-calculus, with lazy evaluation of expressions and many impressive higher-order features.

It is beyond the scope of our study to give any general, yet accurate account of the language. We only overview some of its characteristics and point to Haskell's website `http://haskell.org/` for the most appropriate introduction and further references.

In Chapter 5, we exemplify and illustrate the features of Haskell step by step while developing ElixirFM. In Chapter 9, we present the implementation of a grammar of rewrite rules for Encode Arabic.

5. ElixirFM Design

ElixirFM is a high-level implementation of Functional Arabic Morphology. It reuses and extends the Functional Morphology for Haskell (Forsberg and Ranta, 2004), yet the treatment of the language-specific issues constitutes our original contribution.

Inflection and derivation are modeled in terms of paradigms, grammatical categories, lexemes and word classes. The functional and the structural aspects of morphology are clearly separated. The computation of analysis or generation is conceptually distinguished from the general-purpose linguistic model.

The lexicon of ElixirFM is designed with respect to abstraction, yet is no more complicated than printed dictionaries. It is derived from the open-source Buckwalter lexicon (Buckwalter, 2002) and is enhanced with other unique information.

In Section 5.1, we survey some of the categories of the syntax–morphology interface in Modern Written Arabic, described by Functional Arabic Morphology. In passing, we introduce the basic concepts of programming in Haskell, a modern purely functional language that is an excellent choice for declarative generative modeling of morphologies, as Forsberg and Ranta (2004) have shown.

Section 5.2 is devoted to describing the lexicon of ElixirFM. We develop a so-called domain-specific language embedded in Haskell with which we achieve lexical definitions that are simultaneously a source code that can be checked for consistency, a data structure ready for rather independent processing, and still an easy-to-read-and-edit document resembling the printed dictionaries.

In Section 5.3, we illustrate how rules of inflection and derivation interact with the parameters of the grammar and the lexical information. We claim that the system is reusable in many applications, including computational analysis and generation in various modes, exploring and exporting of the lexicon, printing of the inflectional paradigms, etc.

5.1. Morphosyntactic Categories

Different morphological models categorize individual inflected word forms differently. Some models introduce features and values that are not always complete with respect to the inflectional system, nor mutually orthogonal. We explain what we mean by revisiting the notions of state and definiteness in contemporary written Arabic.

Functional Arabic Morphology refactors the category of state, also denoted as formal definiteness, depending on two parameters. The first controls prefixation of the (virtual) definite article, the other reduces some suffixes if the word is a head of an annexation. In ElixirFM, we define these parameters as type synonyms to standard Haskell types:

```
    type Definite = Maybe Bool
    type Annexing = Bool
```

The `Definite` values include `Just True` for forms with the definite article, `Just False` for forms in some compounds or after *lā* لَا or *yā* يَا (absolute negatives or vocatives), and `Nothing` for forms that reject the definite article for other reasons. The `State` category is in our view considered as a result of coupling the two independent parameters:

```
    type State = Couple Definite Annexing
```

Thus, the indefinite state describes a word void of the definite article(s) and not heading an annexation, i.e. `Nothing :-: False`. Conversely, *ar-rafīʿū* اَلرَّفِيعُو 'the-highs-of' is in the state `Just True :-: True`. The classical construct state is `Nothing :-: True`. The definite state is `Just _ :-: False`, where _ is `True` for (El Dada and Ranta, 2006) and `False` for (Fischer, 2001).

```
|> "k t b" <| [

    FaCaL          `verb`  [ "write", "be destined" ]       `imperf`  FCuL,

    FiCAL          `noun`  [ "book" ]                       `plural`  FuCuL,

    FiCAL |< aT    `noun`  [ "writing" ],

    FiCAL |< aT    `noun`  [ "essay", "piece of writing" ]  `plural`  FiCAL |< At,

    FACiL          `noun`  [ "writer", "author", "clerk" ]  `plural`  FaCaL |< aT
                                                            `plural`  FuCCAL,

    FuCCAL         `noun`  [ "kuttab", "Quran school" ]     `plural`  FaCACIL,

    MaFCaL         `noun`  [ "office", "department" ]       `plural`  MaFACiL,

    MaFCaL |< Iy   `adj`   [ "office" ],

    MaFCaL |< aT   `noun`  [ "library", "bookstore" ]       `plural`  MaFACiL    ]
```

Figure 1. Entries of the ElixirFM lexicon nested under the root k t b كتب using morphophonemic templates.

5.2. ElixirFM Lexicon

Unstructured text is just a list of characters, i.e. a string. Yet words do have structure, particularly in Arabic. We work with strings as the superficial word forms, but the internal representations are more abstract (and computationally more efficient, too).

The definition of *lexemes* can include the derivational *root and pattern* information if appropriate, cf. (Habash, Rambow, and Kiraz, 2005), and our model does encourage this. The surface word *kitāb* كِتَاب ‘book’ can decompose to the triconsonantal root *k t b* كتب and the morphophonemic pattern `FiCAL`:

```
    data PatternT = FaCaL            | FAL      | FaCY     | FaCL
                  | HaFCAL    | HACAL           | HaFCA'              | HACA'
                  | FiCAL                       | FiCA'
                  | FuCCAL    | FUCAL
                  | TaFACuL                     | TaFACI
                  | IFtiCAL          | IFtiyAL  | IFtiCA'
                  | MustaFCaL        | MustaFAL | MustaFCY | MustaFaCL

                  | {- ... -}                         deriving (Eq, Enum, Show)
```

The `deriving` clause associates the type `PatternT` with methods for testing equality, enumerating all the values, and turning the names of the values into strings. Of course, ElixirFM provides functions for properly interlocking the patterns with the roots:

```
merge "k t b"  FiCAL      ⟶    "kitAb"
merge "^g w b" IstaFAL    ⟶    "ista^gAb"
merge "^g w b" IstaFCaL   ⟶    "ista^gwab"
merge "s ' l"  MaFCUL     ⟶    "mas'Ul"
merge "r ' y"  HAFA'      ⟶    "'ArA'"
merge "z h r"  IFtaCaL    ⟶    "izdahar"
```

The *izdahar* اِزْدَهَر 'to flourish' case exemplifies that exceptionless assimilations need not be encoded in the patterns, but can instead be hidden in rules.

The whole generative model adopts the notation of ArabTEX (Lagally, 2004) as a meta-encoding of both the orthography and phonology. Therefore, instantiation of the `"'"` *hamza* carriers or other merely orthographic conventions do not obscure the morphological model. With Encode Arabic interpreting the notation, ElixirFM can at the surface level process the original Arabic script (non-)vocalized to any degree or work with some kind of transliteration or even transcription thereof.

Morphophonemic patterns represent the stems of words. The various kinds of abstract prefixes and suffixes can be expressed either as atomic values, or as literal strings wrapped into extra constructors:

```
data Prefix = Al | LA | Prefix String

data Suffix = Iy | AT | At | An | Ayn | Un | In | Suffix String

al = Al; lA = LA                        -- function synonyms

aT = AT; ayn = Ayn; aN = Suffix "aN"
```

Affixes and patterns are put together via the `Morphs a` data type, where `a` is a triliteral pattern `PatternT` or a quadriliteral `PatternQ` or a non-templatic word stem `Identity` of type `PatternL`:

```
data PatternL = Identity
data PatternQ = KaRDaS | KaRADiS {- ... -}

data Morphs a = Morphs a [Prefix] [Suffix]
```

The word *lā-silkīy* لَاسِلْكِيّ 'wireless' can thus be decomposed as the root *s l k* سلك and the value `Morphs FiCL [LA] [Iy]`. Shunning such concrete representations, we define new operators `>|` and `|<` that denote prefixes, resp. suffixes, inside `Morphs a`. If it is strings that need to be prefixed or suffixed, the shorthand `>>|` and `|<<` can also be used:

```
lA >| FiCL |< Iy    ⟶    Morphs FiCL [LA] [Iy]

al >| lA >| FiCL |< Iy |<< "u"    ⟶    Morphs FiCL [Al, LA] [Suffix "u", Iy]
```

With the introduction of patterns and templates, some synonymous functions and the intuitive operators, we start developing what can be viewed as a domain-specific language embedded in the general-purpose programming language. Encouraged by the flexibility of many other domain-specific languages in Haskell, we design the lexicon to look as shown in Figure 1, yet be a verifiable source code defining a directly interpretable data structure.

```
data Mood = Indicative | Subjunctive | Jussive | Energetic  deriving (Eq, Enum)
data Gender = Masculine | Feminine                          deriving (Eq, Enum)

data ParaVerb = VerbP      Voice Person Gender Number
              | VerbI Mood Voice Person Gender Number
              | VerbC                   Gender Number       deriving Eq


paraVerbC :: Morphing a b => Gender -> Number -> [Char] -> a -> Morphs b
paraVerbC g n i = case n of

        Singular    -> case g of    Masculine ->  prefix i . suffix ""
                                    Feminine  ->  prefix i . suffix "I"

        Plural      -> case g of    Masculine ->  prefix i . suffix "UW"
                                    Feminine  ->  prefix i . suffix "na"

        _           ->                            prefix i . suffix "A"
```

Figure 2. Excerpt from the implementation of verbal inflectional features and paradigms in ElixirFM.

The lexicon of ElixirFM is derived from the open-source Buckwalter lexicon (Buckwalter, 2002). We devised an algorithm in Perl using the morphophonemic patterns of ElixirFM that finds the roots and templates of the lexical items, as they are available only partially in the original, and produces the lexicon in formats for Perl and for Haskell.

5.3. Morphological Rules

Inferential–realizational morphology is modeled in terms of paradigms, grammatical categories, lexemes and word classes. ElixirFM implements the comprehensive rules that draw the information from the lexicon and generate the word forms given the appropriate morphosyntactic parameters. The whole is invoked through a convenient `inflect` method.

The lexicon and the parameters determine the choice of paradigms. The template selection mechanism differs for nominals (providing plurals) and for verbs (providing all needed stem alternations in the extent of the entry specifications of e.g. Hans Wehr's dictionary).

In Figure 2, the algebraic data type `ParaVerb` restricts the space in which verbs are inflected by defining three Cartesian products of the elementary categories: a verb can have `VerbP` perfective forms inflected in voice, person, gender, number, `VerbI` imperfective forms inflected also in mood, and `VerbC` imperatives inflected in gender and number only.

The paradigm for inflecting imperatives, the only such paradigm in ElixirFM, is implemented as `paraVerbC`. It is a function parametrized by some particular value of gender `g` and number `n`, as well as the initial imperative prefix `i` and the verbal stem (both inferred from the morphophonemic patterns in the lexical entry) and yielding the inflected form.

The definition of `paraVerbC` is simple and concise due to the chance to compose with `.` the partially applied `prefix` and `suffix` functions and to virtually omit the next argument. This advanced formulation perhaps does not seem as minimal as the mere specification of the literal endings or prefixes, but we present it here to illustrate the options that there are. An abstract paradigm can be used on more abstract types than just strings. Inflected forms need not be merged with roots yet, and can retain the internal structure:

```
paraVerbC Feminine Plural "u" FCuL    ⟶    "u" >>| FCuL |<< "na"

merge "k t b" (Prefix "u" >| FCuL |< Suffix "na")   ⟶
```

"*uktubna*" *uktubna* اُكْتُبْنَ fem.pl. 'write!'

The highlight of the Arabic morphology is that the 'irregular' inflection actually rests in strictly observing some additional rules, the nature of which is phonological. Therefore, surprisingly, ElixirFM does not even distinguish between verbal and nominal word formation when enforcing these rules. This reduces the number of paradigms to the prototypical 3 verbal and 5 nominal. Yet, the model is efficient.

Nominal inflection is also driven by the information from the lexicon and by phonology. Note that the morphophonemic patterns and the `Morphs a` templates are actually extremely informative. We can use them as determining the inflectional class and the paradigm function, and thus we can almost avoid other unintuitive or excessive indicators of the kind of weak morphology, diptotic inflection, and the like.

5.4. Applications

The ElixirFM linguistic model and the data of the lexicon can be integrated into larger applications or used as standalone libraries and resources.

The language-independent part of the system could rest in the Functional Morphology library (Forsberg and Ranta, 2004). Among other useful things, it implements the compilation of the inflected word forms and their associated morphosyntactic categories into morphological analyzers and generators. The method used for analysis is deterministic parsing with tries, cf. also (Huet, 2002, Ljunglöf, 2002).

Nonetheless, ElixirFM provides an original analysis method exploiting the inflectional invariant defined in Chapter 3. We can, at least in the present version of the implementation, dispense with the compilation into tries, and we use rather minimal computational resources.

We define a class of types that can be `Resolved`, which introduces one rather general method `resolveBy` and one more specific method `resolve` saying that the form in question should be resolved by equality with the inflected forms in the model. The generic `resolveBy` method can be used esp. for recognition of partially vocalized or completely non-vocalized representations of Arabic, or allow in fact arbitrary kinds of omissions, cf. Chapter 6.

Reusing and extending the original Functional Morphology library, ElixirFM also provides functions for exporting and pretty-printing the linguistic model into XML, LaTeX, Perl, SQL, and other custom formats.

6. Other Listings

This chapter is a non-systematic overview of the features of ElixirFM. It can serve as a tutorial for the first sessions with ElixirFM in the environment of the Hugs interpreter. Here, we present just a couple of examples.

```
ElixirFM> inflect (FiCAL `noun` []) "--------2-"

[("N------S2I",[("f ` l",FiCAL |<< "iN")]),("N------S2R",[(......
,("N------D2L",[("f ` l",FiCAL |<< "ay")]),...,("N------P2L",[])]

ElixirFM> pretty $ inflect (RE "k t b" $ FiCAL `noun` []) "-------S2[IDR]"

("N------S2I",[("k t b",FiCAL |<< "iN")])
("N------S2R",[("k t b",FiCAL |<< "i")])
("N------S2D",[("k t b",al >| FiCAL |<< "i")])

ElixirFM> uncurry merge ("k t b", FiCAL |<< "iN")

"kitAbiN"

ElixirFM> pretty $ inflect (RE "k t b" $ FiCAL `noun` [] `plural` FuCuL)
                           "-------P2[IDR]"

("N------P2I",[("k t b",FuCuL |<< "iN")])
("N------P2R",[("k t b",FuCuL |<< "i")])
("N------P2D",[("k t b",al >| FuCuL |<< "i")])

ElixirFM> pretty $ resolveBy (omitting "aiuAUI") "ktbuN"

N------S1I kitAbuN "k t b" FiCAL ["book"]
N------P1I kutubuN "k t b" FiCAL ["book"]
N------S1I kAtibuN "k t b" FACiL ["writer","author","clerk"]
A-----MS1I kAtibuN "k t b" FACiL ["writing"]

ElixirFM> pretty $ resolveBy (omitting $ (encode UCS . decode Tim) "~aiuKNF")
                             (decode Tim "ktAb")

N------S1I kitAbuN "k t b" FiCAL ["book"]
N------S1R kitAbu "k t b" FiCAL ["book"]
N------S1A kitAbu "k t b" FiCAL ["book"]
N------S1L kitAbu "k t b" FiCAL ["book"]
N------S2I kitAbiN "k t b" FiCAL ["book"]
N------S2R kitAbi "k t b" FiCAL ["book"]
N------S2A kitAbi "k t b" FiCAL ["book"]
N------S2L kitAbi "k t b" FiCAL ["book"]
N------S4R kitAba "k t b" FiCAL ["book"]
N------S4A kitAba "k t b" FiCAL ["book"]
N------S4L kitAba "k t b" FiCAL ["book"]
N------P1I kuttAbuN "k t b" FACiL ["writer","author","clerk"]
N------P1R kuttAbu "k t b" FACiL ["writer","author","clerk"]
N------P1A kuttAbu "k t b" FACiL ["writer","author","clerk"]
```

```
N------P1L kuttAbu "k t b" FACiL ["writer","author","clerk"]
N------P2I kuttAbiN "k t b" FACiL ["writer","author","clerk"]
N------P2R kuttAbi "k t b" FACiL ["writer","author","clerk"]
N------P2A kuttAbi "k t b" FACiL ["writer","author","clerk"]
N------P2L kuttAbi "k t b" FACiL ["writer","author","clerk"]
N------P4R kuttAba "k t b" FACiL ["writer","author","clerk"]
N------P4A kuttAba "k t b" FACiL ["writer","author","clerk"]
N------P4L kuttAba "k t b" FACiL ["writer","author","clerk"]
```

7. MorphoTrees

MorphoTrees (Smrž and Pajas, 2004) evolved as an idea of building effective and intuitive hierarchies over the information presented by morphological systems. Such a concept is especially interesting for Arabic and the Functional Arabic Morphology, yet, it is not limited to the language, nor to the formalism, and various extensions are imaginable.

7.1. The MorphoTrees Hierarchy

As an inspiration for the design of the hierarchies, consider the following analyses of the string `fhm` فهم. Some readings will interpret it as just one token related to the notion of 'understanding', but homonymous for several lexical units, each giving many inflected forms, distinct phonologically despite their identical spelling in the ordinary non-vocalized text. Other readings will decompose the string into two co-occurring tokens, the first one, in its non-vocalized form `f` ف, standing for an unambiguous conjunction, and the other one, `hm` هم, analyzed as a verb, noun, or pronoun, each again ambiguous in its functions.

Clearly, this type of concise and 'structured' description does not come ready-made—we have to construct it on top of the overall morphological knowledge. We can take the output solutions of morphological analyzers and process them according to our requirements on tokenization and 'functionality' stated above. Then, we can merge the analyses and their elements into a five-level hierarchy similar to that of Figure 3. The leaves of it are the full forms of the tokens plus their tags as the atomic units. The root of the hierarchy represents the input string, or generally the input entity (some linear or structured subpart of the text). Rising from the leaves up to the root, there is the level of lemmas of the lexical units, the level of non-vocalized canonical forms of the tokens, and the level of decomposition of the entity into a sequence of such forms, which implies the number of tokens and their spelling.

Note that the MorphoTrees hierarchy itself might serve as a framework for evaluating morphological taggers, lemmatizers and stemmers of Arabic, since it allows for resolution of their performance on the different levels, which does matter with respect to the variety of applications.

7.2. MorphoTrees Disambiguation

The annotation of MorphoTrees rests in selecting the applicable sequence of tokens that analyze the entity in the context of the discourse. In a naive setting, an annotator would be left to search the trees by sight, decoding the information for every possible analysis before coming across the right one. If not understood properly, the supplementary levels of the hierarchy would rather tend to be a nuisance …

Instead, MorphoTrees in TrEd take great advantage of the hierarchy and offer the option to restrict one's choice to subtrees and hide those leaves or branches that do not conform to the criteria of the annotation. Moreover, many restrictions are applied automatically, and the decisions about the tree can be controlled in a very rapid and elegant way.

The MorphoTrees of the entity `fhm` فهم in Figure 3 are in fact already annotated. The annotator was expecting, from the context, the reading involving a conjunction. By pressing the shortcut `C` at the root node, he restricted the tree accordingly, and the only one eligible leaf satisfying the `C---------` tag restriction was selected at that moment. Nonetheless, the *fa-* فَ 'so' conjunction is part of a two-token entity, and some annotation of the second token must also be performed. Automatically, all inherited restrictions were removed from the `hm` هم subtree (notice the empty tag in the flag over it), and the subtree unfolded again. The annotator moved the node cursor to the lemma for the pronoun, and restricted its readings to the nominative `--------1-` by pressing another mnemonic shortcut `1`, upon which the single conforming leaf *hum* هُم 'they' was selected automatically. There were no more decisions to make and the annotation proceeded to the next entity of the discourse.

Alternatively, the annotation could be achieved merely by typing `s1`. The restrictions would unambiguously lead to the nominative pronoun, and then, without human intervention, to the other token, the unambiguous conjunction. These automatic decisions need no linguistic model, yet are very effective.

7.3. Further Discussion

Hierarchization of the selection task seems to be the most important contribution of the idea. The suggested meaning of the levels of the hierarchy mirrors the linguistic theory and also one particular strategy for decision-making, neither of which are universal. If we adapt MorphoTrees to other languages or hierarchies, the power of trees remains, though—efficient top-down search or bottom-up restrictions, gradual focusing on the solution, refinement, inheritance and sharing of information, etc.

The levels of MorphoTrees are extensible internally as well as externally in both directions, and the concept incites new views on some issues encompassed by morphological analysis and disambiguation.

In PADT, whose MorphoTrees average roughly 8–10 leaves per entity depending on the data set while the result of annotation is 1.16–1.18 tokens per entity, restrictions as a means of direct access to the solutions improve the speed of annotation significantly.

We propose to evaluate tokenizations in terms of the Longest Common Subsequence problem (cf. (Crochemore et al., 2000)). The tokens that are the members of the LCS with some referential tokenization, are considered correctly recognized. Dividing the length of the LCS by the length of one of the sequences, we get recall, doing it for the other of the sequences, we get precision. The harmonic mean of both is $F_{\beta=1}$-measure (cf. (Manning and Schütze, 1999)).

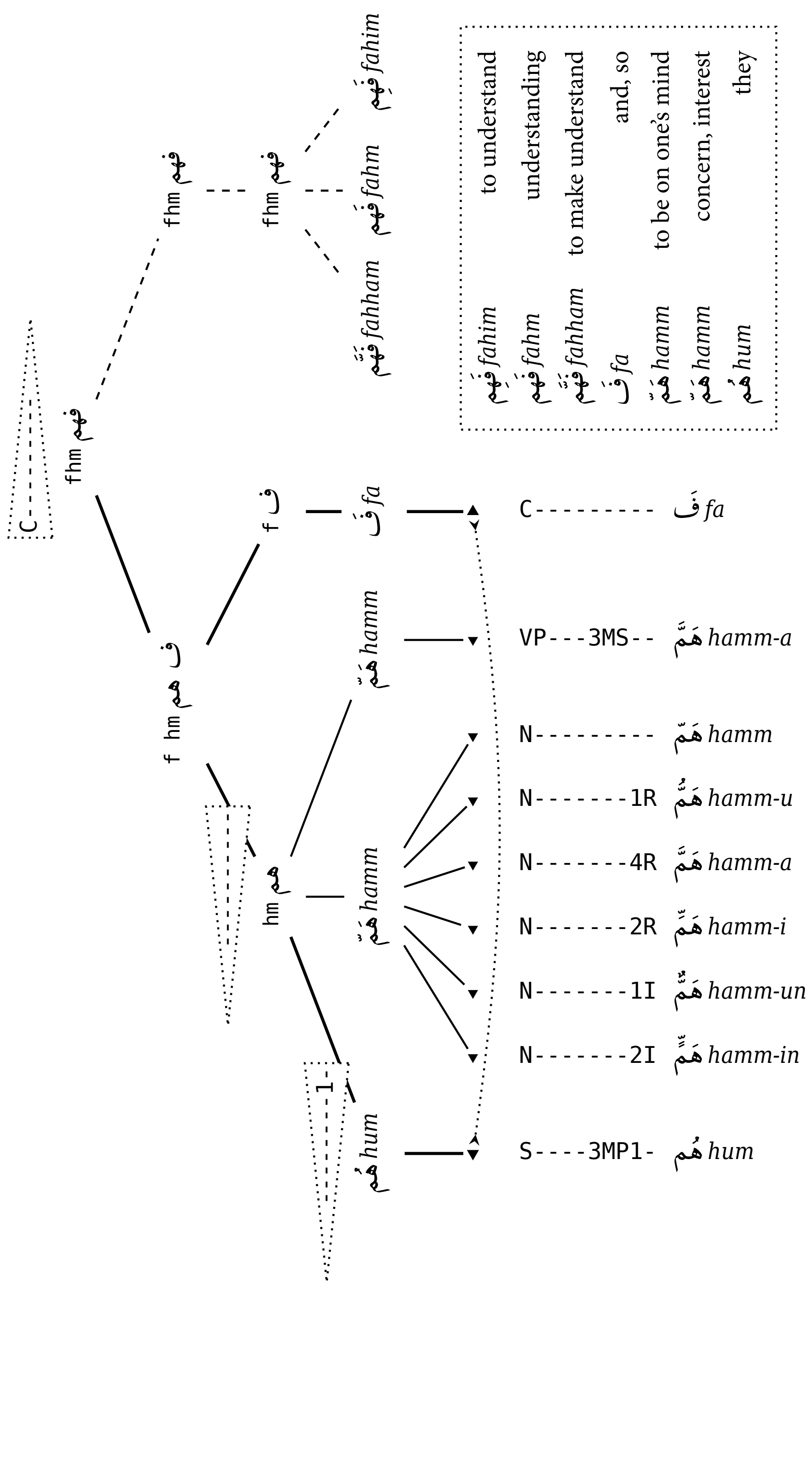


Figure 3. MorphoTrees of the orthographic string fhm فهم *including annotation with restrictions. The dashed lines indicate there is no solution suiting the inherited restrictions in the given subtree. The dotted line symbolizes there might be implicit morphosyntactic constraints between the adjacent tokens in the analyses.*

8. Lexicon versus Treebank

This chapter outlines the structure of linguistic description in the framework of Functional Generative Description and motivates our specific concerns about Arabic within the Prague Arabic Dependency Treebank.

8.1. Functional Description of Language

Prague Arabic Dependency Treebank (Hajič et al., 2004a, Hajič et al., 2004b) is a project of analyzing large amounts of linguistic data in Modern Written Arabic in terms of the formal representation of language that originates in the Functional Generative Description (Sgall, 1967, Sgall, Hajičová, and Panevová, 1986, Panevová, 1980, Hajičová and Sgall, 2003).

In this theory, the formal representation delivers the linguistic meaning of what is expressed by the surface realization, i.e. the natural language. The description is designed to enable generating the natural language out of the formal representations. By constructing the treebank, we provide a resource for computational learning of the correspondences between both languages, the natural and the formal.

Morphological annotations identify the textual forms of a discourse lexically and recognize the morphosyntactic categories that the forms assume. Processing on the analytical level describes the superficial syntactic relations present in the discourse, whereas the tectogrammatical level reveals the underlying structures and restores the linguistic meaning (cf. (Sgall, Panevová, and Hajičová, 2004)).

Linguistic data, i.e. mostly newswire texts in their written form, are gradually analyzed in this system of levels, and their linguistic meaning is thus reconstructed and made explicit.

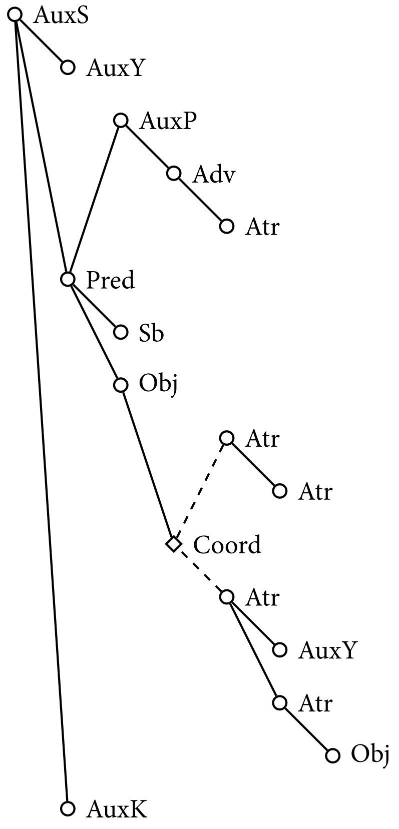
8.2. Analytical Syntax

The tokens with their disambiguated grammatical information enter the annotation of analytical syntax (Žabokrtský and Smrž, 2003, Hajič et al., 2004b). This level is formalized into dependency trees the nodes of which are the tokens. Relations between nodes are classified with analytical syntactic functions. More precisely, it is the whole subtree of a dependent node that fulfills the particular syntactic function with respect to the governing node.

In Figure 4, we analyze a verbal sentence with coordination and a subordinate relative clause. Coordination is depicted with a diamond node and dashed 'dependency' edges between the coordination node and its member coordinants.

Both clauses and nominal expressions can assume the same analytical functions—the attributive clause in our example is Atr, just like in the case of nominal attributes. Pred denotes the main predicate, Sb is subject, Obj is object, Adv stands for adverbial. AuxP, AuxY and AuxK are auxiliary functions of specific kinds.

The coordination relation is different from the dependency relation, but we can depict it in the tree-like manner, too. The coordinative node becomes Coord, and the subtrees that are the members of the coordination are marked as such (cf. dashed edges). Dependents modifying

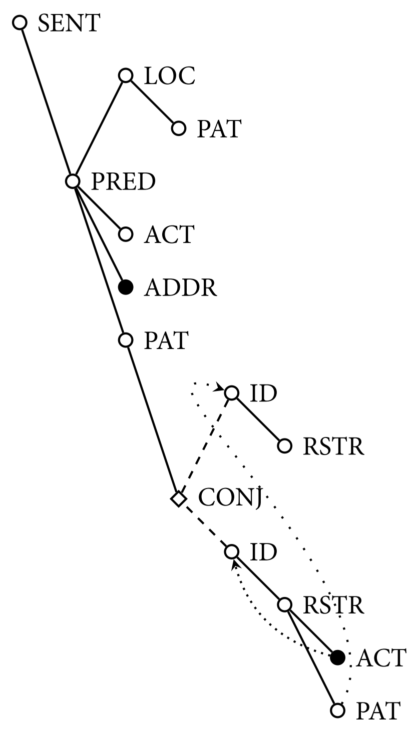


وَ	*wa-*	and	`C---------`
فِي	*fī*	in	`P---------`
مِلَفِّ	*milaffi*	collection/file-of	`N-------2R`
الأَدَبِ	*al-ʾadabi*	the-literature	`N-------2D`
طَرَحَت	*ṭaraḥat*	it-presented	`VP-A-3FS--`
المَجَلَّةُ	*al-maǧallatu*	the-magazine	`N-----FS1D`
قَضِيَّةَ	*qaḍīyata*	issue-of	`N-----FS4R`
اللُّغَةِ	*al-luġati*	the-language	`N-----FS2D`
العَرَبِيَّةِ	*al-ʿarabīyati*	the-Arabic	`A-----FS2D`
وَ	*wa-*	and	`C---------`
الأَخطَارِ	*al-ʾaḫṭāri*	the-dangers	`N-------2D`
الَّتِي	*allatī*	that	`SR----FS--`
تُهَدِّدُ	*tuhaddidu*	they-threaten	`VIIA-3FS--`
هَا	*-hā*	it	`S----3FS4-`
.	.	.	`G---------`

Figure 4. An analytical representation of the sentence In the section on literature, the magazine presented the issue of the Arabic language and the dangers that threaten it.

the coordination as a whole would attach directly to the Coord node, yet would not be marked as coordinants—therefrom, the need for distinguishing coordination and pure dependency in the trees.

The immediate-dominance relation that we capture in the annotation is independent of the linear ordering of words in an utterance, i.e. the linear-precedence relation (Debusmann, 2006). Thus, the expressiveness of the dependency grammar is stronger than that of phrase-structure context-free grammar. The dependency trees can become non-projective by featuring crossing dependencies, which reflects the possibility of relaxing word order while preserving the links of grammatical government.



مِلَفّ *milaff*	collection	Masc.Sing.Def
أَدَب *ʾadab*	literature	Masc.Sing.Def
طَرَح *ṭaraḥ*	to present	Ind.Ant.Act
مَجَلَّة *mağallah*	magazine	Fem.Sing.Def
هُوَ *huwa*	someone	GenPronoun
قَضِيَّة *qaḍīyah*	issue	Fem.Sing.Def
لُغَة *luġah*	language	Fem.Sing.Def
عَرَبِيّ *ʿarabīy*	Arabic	Adjective
وَ *wa-*	and	Coordination
خَطَر *ḫaṭar*	danger	Masc.Plur.Def
هَدَّد *haddad*	to threaten	Ind.Sim.Act
هِيَ *hiya*	it	PersPronoun
هِيَ *hiya*	it	PersPronoun

Figure 5. A tectogrammatical representation of the sentence In the section on literature, the magazine presented the issue of the Arabic language and the dangers that threaten it.

8.3. Tectogrammatics

The analytical syntax is yet a precursor to the deep syntactic annotation (Sgall, Panevová, and Hajičová, 2004, Mikulová and others, 2006) with the following characteristics:

deleted nodes only autosemantic lexemes and coordinative nodes are involved in tectogrammatics; synsemantic lexemes, such as prepositions or particles, are deleted from the trees and may be instead reflected in the values of deep grammatical categories, called *grammatemes*, that are associated with the relevant autosemantic nodes

inserted nodes autosemantic lexemes that do not appear explicitly in the surface syntax, yet that are required as obligatory by valency frames or by other criteria of tectogrammatical well-formedness, are inserted into the deep syntactic structures; the elided lexemes may be copies of other explicit nodes, or may be restored even as generic or unspecified

functors are the tectogrammatical functions describing deep dependency relations; the underlying theory distinguishes *arguments* (inner participants: ACTor, PATient, ADDRessee, ORIGin, EFFect) and *adjuncts* (free modifications, e.g.: LOCation, CAUSe, MANNer, TimeWHEN, ReSTRictive, APPurtenance) and specifies the type of coordination (e.g. CONJunctive, DISJunctive, ADVerSative, ConSeQuential)

grammatemes are the deep grammatical features that are necessary for proper generation of the surface form of an utterance, given the tectogrammatical tree as well, cf. (Hajič et al., 2004b)

coreference pronouns are matched with the lexical mentions they refer to; we distinguish *grammatical* coreference (the coreferent is determined by grammar) and *textual* coreference (otherwise); in Figure 5, the densely dotted arc indicates grammatical coreference, the loosely dotted curve denotes textual coreference

Compare the representations in Figures 4 and 5. Note the differences in the set of nodes actually represented, esp. the restored ADDRessee which is omitted in the surface form of the sentence, but is obligatory in the valency frame of the semantics of the PREDicate.

8.4. Dependency and Inherent vs. Inflectional Properties

Analytical syntax makes the agreement relations more obvious. We can often use those relations to infer information on inherent lexical properties as to gender, number, and humanness, as other words in the relation can, with their inflectional or inferred inherent properties, provide enough constraints.

So this problem is a nice example for constraint programming. Our experiments with the treebank so far have been implemented in Perl, and the inference algorithm was not optimal. Neither was the handling of constraints that (perhaps by an error in the annotation) contradict the other ones. Anyway, we did arrive at promising preliminary results.

These experiments have not been fully completed, though, and their revision is needed. In view of that, we consider formulating the problem in the Mozart/Oz constraint-based programming environment ((Van Roy and Haridi, 2004), chapters 9 and 12).

8.5. Tectogrammatics and Derivational Morphology

We can define default derivations of participles and deverbal nouns, the *maṣdar*s, or consider transformations of patterns between different derivational forms, like in the case of Czech where lexical-semantic shifts are also enforced in the valency theory (cf. (Žabokrtský, 2005)). If the default happens to be inappropriate, then a lexical entry can be extended to optionally include the lexicalized definition of the information that we might require.

The concrete transformations that should apply on the tectogrammatical level are a research in progress, performed by the whole PADT team.

The ability to do the transformations, however, is expected in near future as a direct extension of the ElixirFM system.

9. Encode Arabic

This chapter contains details about the implementations related to processing the ArabTEX notation and its extensions. The mentioned software is open-source and is available via `http://sourceforge.net/projects/encode-arabic/`.

Extending ArabTEX The `alocal` package implements some of the notational extensions of Encode Arabic to work in ArabTEX.

The `acolor` package adds colorful typesetting to ArabTEX. Thanks are due to Karel Mokrý who implemented the core of this functionality originally.

Independent Libraries The Perl implementation of Encode Arabic is documented at `http://search.cpan.org/dist/Encode-Arabic/`.

In the thesis, we present parts of the implementation of our Haskell library for processing the Arabic language in the ArabTEX transliteration (Lagally, 2004), a non-trivial and multi-purpose notation for encoding Arabic orthographies and phonetic transcriptions in parallel. Our approach relies on the Pure Functional Parsing library developed in (Ljunglöf, 2002), which we accommodate to our problem and partly extend. We promote modular design in systems for modeling or processing natural languages.

Conclusion

In the thesis, we developed the theory of Functional Arabic Morphology and designed ElixirFM as its high-level functional and interactive implementation written in Haskell.

Next to numerous theoretical points on the character of Arabic morphology and its relation to syntax, we proposed a model that represents the linguistic data in an abstract and extensible notation that encodes both *orthography* and *phonology*, and whose interpretation is customizable. We developed a domain-specific language in which the lexicon is stored and which allows easy manual editing as well as automatic verification of consistency. We believe that the modeling of both the *written* language and the *spoken* dialects can share the presented methodology.

ElixirFM and its lexicons are licensed under GNU GPL and are available on `http://sourceforge.net/projects/elixir-fm/`. The implementations of Encode Arabic, MorphoTrees, and the ArabTEX extensions are published likewise.

We intend to improve our work further and integrate ElixirFM closely with MorphoTrees as well as with both levels of syntactic representation in the Prague Arabic Dependency Treebank.

Acknowledgement The research for the thesis was supported by the Ministry of Education of the Czech Republic (MSM 0021620838), by the Grant Agency of Charles University in Prague (UK 373/2005), and by the Grant Agency of the Czech Academy of Sciences (1ET101120413).

Bibliography

Al-Sughaiyer, Imad A. and Ibrahim A. Al-Kharashi. 2004. Arabic Morphological Analysis Techniques: A Comprehensive Survey. *Journal of the American Society for Information Science and Technology*, 55(3):189–213.

Badawi, Elsaid, Mike G. Carter, and Adrian Gully. 2004. *Modern Written Arabic: A Comprehensive Grammar*. Routledge.

Baerman, Matthew, Dunstan Brown, and Greville G. Corbett. 2006. *The Syntax-Morphology Interface. A Study of Syncretism*. Cambridge Studies in Linguistics. Cambridge University Press.

Beesley, Kenneth R. 1998. Arabic Morphology Using Only Finite-State Operations. In *COLING-ACL'98 Proceedings of the Workshop on Computational Approaches to Semitic languages*, pages 50–57.

Beesley, Kenneth R. 2001. Finite-State Morphological Analysis and Generation of Arabic at Xerox Research: Status and Plans in 2001. In *EACL 2001 Workshop Proceedings on Arabic Language Processing: Status and Prospects*, pages 1–8, Toulouse, France.

Beesley, Kenneth R. and Lauri Karttunen. 2003. *Finite State Morphology*. CSLI Studies in Computational Linguistics. CSLI Publications, Stanford, California.

Buckwalter, Tim. 2002. Buckwalter Arabic Morphological Analyzer Version 1.0. LDC catalog number LDC2002L49, ISBN 1-58563-257-0.

Cavalli-Sforza, Violetta, Abdelhadi Soudi, and Teruko Mitamura. 2000. Arabic Morphology Generation Using a Concatenative Strategy. In *Proceedings of NAACL 2000*, pages 86–93, Seattle.

Chalabi, Achraf. 2004. Sakhr Arabic Lexicon. In *NEMLAR International Conference on Arabic Language Resources and Tools*, pages 21–24. ELDA.

Crochemore, Maxime, Costas S. Iliopoulos, Yoan J. Pinzon, and James F. Reid. 2000. A Fast and Practical Bit-Vector Algorithm for the Longest Common Subsequence Problem. In *Proceedings of the 11th Australasian Workshop On Combinatorial Algorithms*, Hunter Valley, Australia.

Dada, Ali. 2007. Implementation of the Arabic Numerals and their Syntax in GF. In *ACL 2007 Proceedings of the Workshop on Computational Approaches to Semitic Languages: Common Issues and Resources*, pages 9–16, Prague, Czech Republic, June. Association for Computational Linguistics.

Debusmann, Ralph. 2006. *Extensible Dependency Grammar: A Modular Grammar Formalism Based On Multigraph Description*. Ph.D. thesis, Saarland University.

Ditters, Everhard. 2001. A Formal Grammar for the Description of Sentence Structure in Modern Standard Arabic. In *EACL 2001 Workshop Proceedings on Arabic Language Processing: Status and Prospects*, pages 31–37, Toulouse, France.

El Dada, Ali and Aarne Ranta. 2006. Open Source Arabic Grammars in Grammatical Framework. In *Proceedings of the Arabic Language Processing Conference (JETALA)*, Rabat, Morocco, June 2006. IERA.

El-Sadany, Tarek A. and Mohamed A. Hashish. 1989. An Arabic morphological system. *IBM Systems Journal*, 28(4):600–612.

Fischer, Wolfdietrich. 2001. *A Grammar of Classical Arabic*. Yale Language Series. Yale University Press, third revised edition. Translated by Jonathan Rodgers.

Forsberg, Markus and Aarne Ranta. 2004. Functional Morphology. In *Proceedings of the Ninth ACM SIGPLAN International Conference on Functional Programming, ICFP 2004*, pages 213–223. ACM Press.

Habash, Nizar. 2004. Large Scale Lexeme Based Arabic Morphological Generation. In *JEP-TALN 2004, Session Traitement Automatique de l'Arabe*, Fes, Morocco, April 2004.

Habash, Nizar, Owen Rambow, and George Kiraz. 2005. Morphological Analysis and Generation for Arabic Dialects. In *Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages*, pages 17–24, Ann Arbor, Michigan. Association for Computational Linguistics.

Hajič, Jan, Otakar Smrž, Tim Buckwalter, and Hubert Jin. 2005. Feature-Based Tagger of Approximations of Functional Arabic Morphology. In *Proceedings of the Fourth Workshop on Treebanks and Linguistic Theories (TLT 2005)*, pages 53–64, Barcelona, Spain.

Hajič, Jan, Otakar Smrž, Petr Zemánek, Petr Pajas, Jan Šnaidauf, Emanuel Beška, Jakub Kráčmar, and Kamila Hassanová. 2004a. Prague Arabic Dependency Treebank 1.0. LDC catalog number LDC2004T23, ISBN 1-58563-319-4.

Hajič, Jan, Otakar Smrž, Petr Zemánek, Jan Šnaidauf, and Emanuel Beška. 2004b. Prague Arabic Dependency Treebank: Development in Data and Tools. In *NEMLAR International Conference on Arabic Language Resources and Tools*, pages 110–117. ELDA.

Hajičová, Eva and Petr Sgall. 2003. Dependency Syntax in Functional Generative Description. In *Dependenz und Valenz – Dependency and Valency*, volume I. Walter de Gruyter, pages 570–592.

Holes, Clive. 2004. *Modern Arabic: Structures, Functions, and Varieties.* Georgetown Classics in Arabic Language and Linguistics. Georgetown University Press.

Huet, Gérard. 2002. The Zen Computational Linguistics Toolkit. ESSLLI Course Notes, FoLLI, the Association of Logic, Language and Information.

Kay, Martin. 1987. Nonconcatenative Finite-State Morphology. In *Proceedings of the Third Conference of the European Chapter of the ACL (EACL-87)*, pages 2–10, Copenhagen, Denmark. ACL.

Kiraz, George Anton. 2001. *Computational Nonlinear Morphology with Emphasis on Semitic Languages.* Studies in Natural Language Processing. Cambridge University Press.

Lagally, Klaus. 1992. ArabTEX: Typesetting Arabic with Vowels and Ligatures. In *EuroTEX 92*, page 20, Prague, Czechoslovakia, September 14–18.

Lagally, Klaus. 2004. ArabTEX: Typesetting Arabic and Hebrew, User Manual Version 4.00. Technical Report 2004/03, Fakultät Informatik, Universität Stuttgart, March 11.

Ljunglöf, Peter. 2002. *Pure Functional Parsing. An Advanced Tutorial.* Licenciate thesis, Göteborg University & Chalmers University of Technology.

Manning, Christopher D. and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing.* MIT Press, Cambridge.

McCarthy, John and Alan Prince. 1990. Foot and Word in Prosodic Morphology: The Arabic Broken Plural. *Natural Language and Linguistic Theory*, 8:209–283.

McCarthy, John J. 1981. A Prosodic Theory of Nonconcatenative Morphology. *Linguistic Inquiry*, 12:373–418.

Mikulová, Marie et al. 2006. A Manual for Tectogrammatical Layer Annotation of the Prague Dependency Treebank. Technical report, Charles University in Prague.

Nelken, Rani and Stuart M. Shieber. 2005. Arabic Diacritization Using Finite-State Transducers. In *Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages*, pages 79–86, Ann Arbor.

Othman, Eman, Khaled Shaalan, and Ahmed Rafea. 2003. A Chart Parser for Analyzing Modern Standard Arabic Sentence. In *Proceedings of the MT Summit IX Workshop on Machine Translation for Semitic Languages: Issues and Approaches*, pages 37–44.

Panevová, Jarmila. 1980. *Formy a funkce ve stavbě české věty [Forms and Functions in the Structure of the Czech Sentence]*. Academia.

Ramsay, Allan and Hanady Mansur. 2001. Arabic morphology: a categorial approach. In *EACL 2001 Workshop Proceedings on Arabic Language Processing: Status and Prospects*, pages 17–22, Toulouse, France.

Sgall, Petr. 1967. *Generativní popis jazyka a česká deklinace [Generative Description of Language and Czech Declension]*. Academia.

Sgall, Petr, Eva Hajičová, and Jarmila Panevová. 1986. *The Meaning of the Sentence in Its Semantic and Pragmatic Aspects*. D. Reidel & Academia.

Sgall, Petr, Jarmila Panevová, and Eva Hajičová. 2004. Deep Syntactic Annotation: Tectogrammatical Representation and Beyond. In *HLT-NAACL 2004 Workshop: Frontiers in Corpus Annotation*, pages 32–38. Association for Computational Linguistics.

Smrž, Otakar. 2007. *Functional Arabic Morphology. Formal System and Implementation.* Ph.D. thesis, Charles University in Prague.

Smrž, Otakar and Petr Pajas. 2004. MorphoTrees of Arabic and Their Annotation in the TrEd Environment. In *NEMLAR International Conference on Arabic Language Resources and Tools*, pages 38–41. ELDA.

Stump, Gregory T. 2001. *Inflectional Morphology. A Theory of Paradigm Structure.* Cambridge Studies in Linguistics. Cambridge University Press.

Van Roy, Peter and Seif Haridi. 2004. *Concepts, Techniques, and Models of Computer Programming.* MIT Press, Cambridge, March.

Yaghi, Jim and Sane Yagi. 2004. Systematic Verb Stem Generation for Arabic. In *COLING 2004 Computational Approaches to Arabic Script-based Languages*, pages 23–30, Geneva, Switzerland.

Žabokrtský, Zdeněk. 2005. *Valency Lexicon of Czech Verbs.* Ph.D. thesis, Charles University in Prague.

Žabokrtský, Zdeněk and Otakar Smrž. 2003. Arabic Syntactic Trees: from Constituency to Dependency. In *EACL 2003 Conference Companion*, pages 183–186, Budapest, Hungary.