Functional Arabic Morphology
Principles of Design

Otakar Smrž

Institute of Formal and Applied Linguistics
Faculty of Mathematics and Physics
Charles University in Prague

Prague, November 6, 2006
He will notify them about that through SMS messages, the Internet, and other means.

سيخبرهم بذلك عن طريق الرسائل القصيرة والإنترنت وغيرها.
He will notify them about that through SMS messages, the Internet, and other means.

<table>
<thead>
<tr>
<th>String</th>
<th>Token</th>
<th>Token Tag</th>
<th>Buckwalter’s M-Tags</th>
<th>Token Form</th>
<th>Token Gloss</th>
</tr>
</thead>
<tbody>
<tr>
<td>سيخبرهم</td>
<td>F-----</td>
<td>FUT</td>
<td></td>
<td>sa-</td>
<td>will</td>
</tr>
<tr>
<td>.onNext طريق الرسائل القصيرة والإنترنت وغيرها</td>
<td>VIDA-3MS--</td>
<td>IV3MS+IV+IVSUFF_MOOD:I</td>
<td>yu-ḥbir-u</td>
<td>he-notify</td>
<td></td>
</tr>
<tr>
<td></td>
<td>S-----</td>
<td>3MP4-</td>
<td>IVSUFF_DO:3MP</td>
<td>-hum</td>
<td>them</td>
</tr>
<tr>
<td>بذلك</td>
<td>P-----</td>
<td>PREP</td>
<td></td>
<td>bi-</td>
<td>about/by</td>
</tr>
<tr>
<td>عن طريق الرسائل القصيرة والإنترنت وغيرها</td>
<td>SD-----</td>
<td>MS--</td>
<td>DEM_PRON_MS</td>
<td>ʾan</td>
<td>by/about</td>
</tr>
<tr>
<td></td>
<td>P-----</td>
<td>PREP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>N-----</td>
<td>2R</td>
<td>NOUN+CASE_DEF_GEN</td>
<td>ṭariq-i</td>
<td>way-of</td>
</tr>
<tr>
<td></td>
<td>N-----</td>
<td>2D</td>
<td>DET+NOUN+CASE_DEF_GEN</td>
<td>ar-rasāʾil-i</td>
<td>the-messages</td>
</tr>
<tr>
<td></td>
<td>A-----</td>
<td>FS2D</td>
<td>DET+ADJ+NSUFF_FEM_SG+ +CASE_DEF_GEN</td>
<td>al-qaṣir-at-i</td>
<td>the-short</td>
</tr>
<tr>
<td></td>
<td>C-----</td>
<td>CONJ</td>
<td></td>
<td>wa-</td>
<td>and</td>
</tr>
<tr>
<td></td>
<td>Z-----</td>
<td>2D</td>
<td>DET+NOUN_PROP+       +CASE_DEF_GEN</td>
<td>al-internet-i</td>
<td>the-internet</td>
</tr>
<tr>
<td></td>
<td>C-----</td>
<td>CONJ</td>
<td></td>
<td>wa-</td>
<td>and</td>
</tr>
<tr>
<td></td>
<td>FN----</td>
<td>2R</td>
<td>NEG_PART+CASE_DEF_GEN</td>
<td>ʾgayr-i</td>
<td>other/not-of</td>
</tr>
<tr>
<td></td>
<td>S-----</td>
<td>3FS2-</td>
<td>POSS_PRON_3FS</td>
<td>-hā</td>
<td>them</td>
</tr>
</tbody>
</table>
He will notify them about that through SMS messages, the Internet, and other means.

Sixbirhum b'dallak 'an 'arrij al-rasail al-qasira wal-internett w'ghirah.

<table>
<thead>
<tr>
<th>String</th>
<th>Token</th>
<th>Token Tag</th>
<th>Buckwalter’s M-Tags</th>
<th>Token Form</th>
<th>Token Gloss</th>
</tr>
</thead>
<tbody>
<tr>
<td>سيخبرهم</td>
<td>F------</td>
<td>FUT</td>
<td></td>
<td>sa-</td>
<td>will</td>
</tr>
<tr>
<td>سياخیرهم</td>
<td>VI1A-3MS--</td>
<td>IV3MS+IV+IVSUFF_MOOD:I</td>
<td></td>
<td>yu-ḥbir-u</td>
<td>he-notify</td>
</tr>
<tr>
<td>بهذلک</td>
<td>S------</td>
<td>3MP4-</td>
<td>IVSUFF_DO:3MP</td>
<td>-hum</td>
<td>them</td>
</tr>
<tr>
<td>عن طريق الرسائل القصيرة والإنترنت وغيرها</td>
<td>P------</td>
<td>PREP</td>
<td></td>
<td>bi-</td>
<td>about/by</td>
</tr>
<tr>
<td>عن طريق الرسائل القصيرة والإنترنت وغيرها</td>
<td>SD------</td>
<td>DEM_PRON_MS</td>
<td></td>
<td>dālika</td>
<td>that</td>
</tr>
<tr>
<td>عن طريق الرسائل القصيرة والإنترنت وغيرها</td>
<td>P------</td>
<td>PREP</td>
<td></td>
<td>'an</td>
<td>by/about</td>
</tr>
<tr>
<td>عن طريق الرسائل القصيرة والإنترنت وغيرها</td>
<td>N-------</td>
<td>NOUN+CASE_DEF_GEN</td>
<td></td>
<td>ṣāriq-i</td>
<td>way-of</td>
</tr>
<tr>
<td>عن طريق الرسائل القصيرة والإنترنت وغيرها</td>
<td>N-------</td>
<td>DET+NOUN+CASE_DEF_GEN</td>
<td></td>
<td>ar-rasa'il-i</td>
<td>the-messages</td>
</tr>
<tr>
<td>عن طريق الرسائل القصيرة والإنترنت وغيرها</td>
<td>A-------</td>
<td>DET+ADJ+NSuff_FEM_SG+ +CASE_DEF_GEN</td>
<td></td>
<td>al-qaṣir-at-i</td>
<td>the-short</td>
</tr>
<tr>
<td>عن طريق الرسائل القصيرة والإنترنت وغيرها</td>
<td>C-------</td>
<td>CONJ</td>
<td></td>
<td>wa-</td>
<td>and</td>
</tr>
<tr>
<td>عن طريق الرسائل القصيرة والإنترنت وغيرها</td>
<td>Z-------</td>
<td>DET+NOUN_PROP+ +CASE_DEF_GEN</td>
<td></td>
<td>al-internet-i</td>
<td>the-internet</td>
</tr>
<tr>
<td>عن طريق الرسائل القصيرة والإنترنت وغيرها</td>
<td>C-------</td>
<td>CONJ</td>
<td></td>
<td>wa-</td>
<td>and</td>
</tr>
<tr>
<td>عن طريق الرسائل القصيرة والإنترنت وغيرها</td>
<td>FN-------</td>
<td>NEG_PART+CASE_DEF_GEN</td>
<td></td>
<td>ḡayr-i</td>
<td>other/not-of</td>
</tr>
<tr>
<td>عن طريق الرسائل القصيرة والإنترنت وغيرها</td>
<td>S------</td>
<td>POSS_PRON_3FS</td>
<td></td>
<td>-ḥā</td>
<td>them</td>
</tr>
</tbody>
</table>

Otkar Smrž (Charles University)  Functional Arabic Morphology  November 6, 2006  2 / 26
He will notify them about that through SMS messages, the Internet, and other means.

**String** | **Token** | **Token Tag** | **Buckwalter’s M-Tags** | **Token Form** | **Token Gloss**
--- | --- | --- | --- | --- | ---

سيخبرهم | F | FUT | | sa- | will

will notify | V11A-3MS-- | IV3MS+IV+IVSUFF_MOOD:I | yu-ḥbir-u | he-notify

them | S--3MP4- | IVSUFF_DO:3MP | -hum | them

about | P-- | PREP | bi- | about/by

by/about | SD--MS-- | DEM_PRON_MS | ḍālika | that

way-of | P-- | PREP | ṣan | by/about

the-messages | N-----2R | NOUN+CASE_DEF_GEN | ṣaṭarīq-i | the-short

the-short | N-----2D | DET+NOUN+CASE_DEF_GEN | al-qasīr-at-i | the-short

and | A--FS2D | DET+ADJ+NSUFF_FEM_SG+ +CASE_DEF_GEN | wa- | and

the-internet | Z---2D | DET+NOUN_PROP+ +CASE_DEF_GEN | al-internet-i | the-internet

and | C----- | CONJ | wa- | and

other/not-of | FN--2R | NEG_PART+CASE_DEF_GEN | ḡayr-i | other/not-of

them | S--3FS2- | POSS_PRON_3FS | -ḥā | them
Outline

1 Introduction
Outline

1. Introduction
2. Morphological Theory
   - Incremental vs. Realizational
   - Lexical vs. Inferential
   - Functional Arabic Morphology
1. Introduction
2. Morphological Theory
   - Incremental vs. Realizational
   - Lexical vs. Inferential
   - Functional Arabic Morphology
3. Implementation Design
   - ElixirFM
   - Paradigms, parameters, ...
   - Elixir Lexicon
   - FM Generic
Outline

1. Introduction
2. Morphological Theory
   - Incremental vs. Realizational
   - Lexical vs. Inferential
   - Functional Arabic Morphology
3. Implementation Design
   - ElixirFM
   - Paradigms, parameters, ...
   - Elixir Lexicon
   - FM Generic
4. Extensions
   - Encode Arabic
   - MorphoTrees
Outline

1. Introduction
2. Morphological Theory
   - Incremental vs. Realizational
   - Lexical vs. Inferential
   - Functional Arabic Morphology
3. Implementation Design
   - ElixirFM
   - Paradigms, parameters, ...
   - Elixir Lexicon
   - FM Generic
4. Extensions
   - Encode Arabic
   - MorphoTrees
5. References
Inflectional Morphology

Morphological theories can be classified along two dimensions (Stump 2001).

lexical association of word’s morphosyntactic properties with affixes
Inflectional Morphology

Morphological theories can be classified along two dimensions (Stump 2001).

**lexical** association of word’s morphosyntactic properties with affixes

**inferential** inflection is a result of operations on lexemes; morphosyntactic properties are expressed by the rules that relate the form in a given paradigm to the lexeme
Morphological theories can be classified along two dimensions (Stump 2001).

**lexical** association of word’s **morphosyntactic properties** with **affixes**

**inferential** inflection is a result of **operations** on **lexemes**; morphosyntactic properties are expressed by the **rules** that relate the form in a given **paradigm** to the lexeme

**incremental** words **acquire** morphosyntactic properties only in connection with acquiring the **inflectional exponents** of those properties
Morphological theories can be classified along two dimensions (Stump 2001).

**lexical** association of word’s morphosyntactic properties with affixes

**inferential** inflection is a result of operations on lexemes; morphosyntactic properties are expressed by the rules that relate the form in a given paradigm to the lexeme

**incremental** words acquire morphosyntactic properties only in connection with acquiring the inflectional exponents of those properties

**realizational** association of a set of properties with a word licenses the introduction of the exponents into the word’s morphology
Inflectional Morphology

Morphological theories can be classified along two dimensions (Stump 2001).

- **lexical**: association of word’s morphosyntactic properties with affixes

- **inferential**: inflection is a result of operations on lexemes; morphosyntactic properties are expressed by the rules that relate the form in a given paradigm to the lexeme

- **incremental**: words acquire morphosyntactic properties only in connection with acquiring the inflectional exponents of those properties

- **realizational**: association of a set of properties with a word licenses the introduction of the exponents into the word’s morphology

Evidence favoring inferential–realizational theories over the others is given.
Extended Exponence

The morphosyntactic properties associated with an inflected word may exhibit extended exponence in that word’s morphology. (Stump 2001:4)
The morphosyntactic properties associated with an inflected word may exhibit extended exponence in that word's morphology. (Stump 2001:4)

sa- will
yu-ḥbir-u he-notify
-hum them
Extended Exponence

The morphosyntactic properties associated with an inflected word may exhibit extended exponence in that word’s morphology. (Stump 2001:4)
Underdetermination

The morphosyntactic properties associated with an inflected word’s individual inflectional markings may underdetermine the properties associated with the word as a whole. (Stump 2001:7)
The morphosyntactic properties associated with an inflected word’s individual inflectional markings may underdetermine the properties associated with the word as a whole. (Stump 2001:7)
Underdetermination

The morphosyntactic properties associated with an inflected word’s individual inflectional markings may underdetermine the properties associated with the word as a whole. (Stump 2001:7)
The morphosyntactic properties associated with an inflected word’s individual inflectional markings may underdetermine the properties associated with the word as a whole. (Stump 2001:7)
There is no theoretically significant difference between concatenative and nonconcatenative inflection. (Stump 2001:9)
There is no theoretically significant difference between concatenative and nonconcatenative inflection. (Stump 2001:9)

أخبار عبارة عن إبلاغ

اُهْبَرُ-ا

F--------- FUT

sa-

will

سيخبارهم

VIIA-3MS-- IV3MS+IV+IVSUFF_MOOD:I

yu-ُنبير-ع

he-notify

S-----3MP4- IVSUFF_DO:3MS

-hum

them

رسالة

رِسَالَة 

N--------2D DET+NOUN+CASE_DEF_GEN

ar-*رِسَالَة-

the-messages
There is no theoretically significant difference between concatenative and nonconcatenative inflection. (Stump 2001:9)

أخبر  aḥbar-a  to notify

F--------- FUT  sa-  will
سيخبرهم  VīIa-3MS-- IV3MS+IV+IVSUFF_MOOD:I  yu-ḥbir-u  he-notify
S-----3MP4- IVSUFF_DO:3MS  -hum  them

رسالة  risāl-at-un  a message

رسائل  ar-rasā’il-i  the-messages
Unmotivated Choice

*Exponence is the only association between inflectional markings and morphosyntactic properties.* (Stump 2001:11)

\[ IV_{3MS+IV} IV_{SUFF\_MOOD:1} \] ?? \[ IV_{3MS+IV} IV_{SUFF\_MOOD:1} \] \( yu-hbir-u \)
Unmotivated Choice

Exponence is the only association between inflectional markings and morphosyntactic properties. (Stump 2001:11)

IV3MS+IV+IVSUFF_MOOD:I  ??  IV3MS+IV+IVSUFF_MOOD:I  yu-ḥbir-u

An uncompounded word’s morphological form is not distinct from its phonological form. (Stump 2001:12)

DET+ADJ+NSUFF_FEM_SG+CASE_DEF_GEN  (al-(qaṣīr-at))-i  ??  ((al-qaṣīr)-at)-i
Most computational models of Arabic morphology are lexical in nature. As they are not designed in connection with any syntax–morphology interface, their interpretation is destined to be incremental.
Most computational models of Arabic morphology are **lexical** in nature. As they are not designed in connection with any **syntax–morphology interface**, their interpretation is destined to be **incremental**.

**Functional Arabic Morphology** endorses the **inferential–realizational** views.
Most computational models of Arabic morphology are \textit{lexical} in nature. As they are not designed in connection with any \textit{syntax–morphology interface}, their interpretation is destined to be \textit{incremental}. 

\textbf{Functional Arabic Morphology} endorses the \textit{inferential–realizational} views.

It re-establishes the \textit{system} of \textit{inflectional} and \textit{inherent} morphosyntactic properties and distinguishes precisely the \textit{senses} of their use in the grammar.
Most computational models of Arabic morphology are lexical in nature. As they are not designed in connection with any syntax–morphology interface, their interpretation is destined to be incremental.

Functional Arabic Morphology endorses the inferential–realizational views. It re-establishes the system of inflectional and inherent morphosyntactic properties and distinguishes precisely the senses of their use in the grammar.

Definition of lexemes can include the derivational root and pattern information if appropriate. Modeling of the written language as well as spoken dialects is expected to be methodologically identical.
Outline

1. Introduction
2. Morphological Theory
   - Incremental vs. Realizational
   - Lexical vs. Inferential
   - Functional Arabic Morphology
3. Implementation Design
   - ElixirFM
   - Paradigms, parameters, . . .
   - Elixir Lexicon
   - FM Generic
4. Extensions
   - Encode Arabic
   - MorphoTrees
5. References
ElixirFM is a high-level implementation of Functional Arabic Morphology.
ElixirFM is a high-level implementation of Functional Arabic Morphology.

ElixirFM uses the Functional Morphology library for Haskell and extends it.
ElixirFM is a high-level implementation of Functional Arabic Morphology.

ElixirFM uses the Functional Morphology library for Haskell and extends it.

Morphology is modeled in terms of paradigms, grammatical categories, lexemes and word classes. The computation of analysis or generation is conceptually distinguished from the general-purpose linguistic model.
ElixirFM is a high-level implementation of Functional Arabic Morphology.

ElixirFM uses the Functional Morphology library for Haskell and extends it.

Morphology is modeled in terms of paradigms, grammatical categories, lexemes and word classes. The computation of analysis or generation is conceptually distinguished from the general-purpose linguistic model.

The lexicon of ElixirFM is derived from the open-source Buckwalter lexicon and from the PADT annotations. It is redesigned in important respects.
Paradigms

A \textit{paradigm function} is a function which, when applied to the root of a lexeme $L$ paired with a set of morphosyntactic properties appropriate to $L$, determines the \textit{word form} occupying the corresponding cell in $L$’s paradigm.  

(Stump 2001:32)
A *paradigm function* is a function which, when applied to the root of a lexeme $L$ paired with a set of morphosyntactic properties appropriate to $L$, determines the word form occupying the corresponding cell in $L$’s paradigm. (Stump 2001:32)

```
paradigm :: (Lexeme, Properties) -> WordForm
paradigm (l, ps) = ...
```
Paradigms

A paradigm function is a function which, when applied to the root of a lexeme L paired with a set of morphosyntactic properties appropriate to L, determines the word form occupying the corresponding cell in L’s paradigm. (Stump 2001:32)

```
paradigm :: (Lexeme, Properties) -> WordForm
paradigm (l, ps) = ...

paradigm' :: Lexeme -> Properties -> WordForm
paradigm' l ps = paradigm (l, ps)
paradigm' l ps = (curry paradigm) l ps
paradigm' = curry paradigm

curry :: ((a, b) -> c) -> a -> b -> c
curry f x y = f (x, y)
```
Paradigms

A paradigm function is a function which, when applied to the root of a lexeme \( L \) paired with a set of morphosyntactic properties appropriate to \( L \), determines the word form occupying the corresponding cell in \( L \)'s paradigm. (Stump 2001:32)

\[
\text{paradigm} :: (\text{Lexeme}, \text{Properties}) \rightarrow \text{WordForm} \\
\text{paradigm} \ (l, \ ps) = \ldots
\]

\[
\text{paradigm}' :: \text{Lexeme} \rightarrow \text{Properties} \rightarrow \text{WordForm} \\
\text{paradigm}' \ l \ ps = \text{paradigm} \ (l, \ ps) \\
\text{paradigm}' \ l \ ps = (\text{curry paradigm}) \ l \ ps \\
\text{paradigm}' = \text{curry paradigm}
\]

\[
\text{curry} :: ((a, b) \rightarrow c) \rightarrow a \rightarrow b \rightarrow c \\
\text{curry} \ f \ x \ y = f \ (x, \ y)
\]
Paradigms

A *paradigm function* is a function which, when applied to the root of a lexeme \( L \) paired with a set of morphosyntactic properties appropriate to \( L \), determines the word form occupying the corresponding cell in \( L \)'s paradigm. (Stump 2001:32)

\[
\text{paradigm} \, :: \, (\text{Lexeme}, \text{Properties}) \rightarrow \text{WordForm} \\
\text{paradigm} \, (l, \, ps) = \ldots
\]

\[
\text{paradigm'} \, :: \, \text{Lexeme} \rightarrow \text{Properties} \rightarrow \text{WordForm} \\
\text{paradigm'} \, l \, ps \, = \, \text{paradigm} \, (l, \, ps) \\
\text{paradigm'} \, l \, ps \, = \, (\text{curry paradigm}) \, l \, ps \\
\text{paradigm'} \, = \, \text{curry paradigm}
\]

\[
\text{curry} \, :: \, ((a, \, b) \rightarrow \, c) \rightarrow \, a \rightarrow \, b \rightarrow \, c \\
\text{curry} \, f \, x \, y \, = \, f \, (x, \, y)
\]
Parameters

Instead of feature–value pairs for encoding the morphosyntactic properties (Stump 2001), we use enumerated values of distinct types. The use of data types is essential in the system (Forsberg and Ranta 2004).
Parameters

Instead of feature–value pairs for encoding the morphosyntactic properties (Stump 2001), we use enumerated values of distinct types. The use of data types is essential in the system (Forsberg and Ranta 2004).

```haskell
data Person = First | Second | Third
  deriving (Eq, Enum)
```
Parameters

Instead of feature–value pairs for encoding the morphosyntactic properties (Stump 2001), we use enumerated values of distinct types. The use of data types is essential in the system (Forsberg and Ranta 2004).

```haskell
data Person  = First | Second | Third
deriving (Eq, Enum)

data Mood    = Indicative | Subjunctive | Jussive | Energetic
deriving (Eq, Show, Enum)
```
Parameters

Instead of feature–value pairs for encoding the morphosyntactic properties (Stump 2001), we use enumerated values of distinct types. The use of data types is essential in the system (Forsberg and Ranta 2004).

```haskell
data Person = First | Second | Third
deriving (Eq, Enum)

data Mood = Indicative | Subjunctive
    | Jussive | Energetic
deriving (Eq, Show, Enum)

data ParaVerb = VerbP Voice Person Gender Number
    | VerbI Mood Voice Person Gender Number
    | VerbC Gender Number
deriving Eq
```
representation of the linguistic data in an abstract and extensible notation that encodes both orthography and phonology, and whose interpretation is customizable
Elixir Lexicon

(a) representation of the linguistic data in an abstract and extensible notation that encodes both orthography and phonology, and whose interpretation is customizable

(b) organization of the lexicon so that there is preferably no duplication of information and so that the lexicon can possibly be divided into separate units, as well as be interlinked with external modules
Elixir Lexicon

(a) representation of the linguistic data in an abstract and extensible notation that encodes both orthography and phonology, and whose interpretation is customizable

(b) organization of the lexicon so that there is preferably no duplication of information and so that the lexicon can possibly be divided into separate units, as well as be interlinked with external modules

(c) definition of such format of the lexicon so that editing and understanding the data is not inappropriately difficult, and using such data markup whose syntax is either lightweight, or can be edited/verified with some automatic tools, or both
The linguistic model and the data of the lexicon can be compiled into run-time applications or used as standalone libraries and resources.
The linguistic model and the data of the lexicon can be compiled into runtime applications or used as standalone libraries and resources.

FM Generic implements the compilation of morphological analyzers and generators (Forsberg and Ranta 2004). The method used for analysis is deterministic parsing with tries (Ljunglöf 2002).
The linguistic model and the data of the lexicon can be compiled into run-time applications or used as standalone libraries and resources.

FM Generic implements the compilation of morphological analyzers and generators (Forsberg and Ranta 2004). The method used for analysis is deterministic parsing with tries (Ljunglöf 2002).

FM Generic also provides functions for exporting and pretty-printing the linguistic model into XFST, Lexc, SQL, XML, LaTeX, ...
Outline

1. Introduction
2. Morphological Theory
   - Incremental vs. Realizational
   - Lexical vs. Inferential
   - Functional Arabic Morphology
3. Implementation Design
   - ElixirFM
   - Paradigms, parameters, ...
   - Elixir Lexicon
   - FM Generic
4. Extensions
   - Encode Arabic
   - MorphoTrees
5. References
يُولد جمِيع النُّاس أُحُراراً متساوِينَ في الكرامة وَالحُقُوقِ. وقد وهبوا عَقْلاً وَضَميّاً وَعَلِيمِينَ
أن يُعامل بعضهم بعضًا بروح الأخِاء.

ywld jmyE AlnAs OHrArA mtsAwyn fy AlkrAmp wAlHqwq. wqd
whbwA EqlA wDmyrA wElyhm On yEAml bEDhm bEDA brwH AlIxA’.
يُولد جميع الناس أحرارًا متساوين في الكرامة وحقوقهم. وقد وُهِبوا عقولًا وضَمائرًا وعلَمهم أن يُعامل بعضهم بعضًا بروحٍ الإخاء.

ولد جميع الناس أحرارًا متساوين في الكرامة والحقوق. وقد وُهِبوا عقولًا وضَمائرًا وعلَمهم أن يُعامل بعضهم بعضًا بروح الإخاء.

Yūladu ḡamīʿu 'n-nāsi ḍahrāran mutasāwīna fī 'l-kaɾāmați wa-ʾl-ḥuqūqi. Wa-qad wuḥibū ʿaqlan wa-ḍamīran wa-ʿalāyhim ʾan yūʿāmla baḍūhum baḍan bi-rūḥī 'l-riḥāʾi.

\cap yUladu ḡamIʿu an-nAsi ṣa.hrAraN mutaSAwIna fI al-kaarAmaTI wa-al-.huqUqi.
\cap wa-qad wuhibUA ʿaqlaN wa-.damIraN wa-ʿalāyhim ʿan yuʿAmila baʿ.duhum baʿ.daN bi-rU.hi al-ʾi_hAʿi.
Extensions
Encode Arabic

**Encode Arabic**

biruwHi {loIixaA’i ← بِرُوحِ آْلاْخَاهِ ← bi-rU.hi al-’i_hA’i

Implemented in **Perl** and available on CPAN as **Encode-Arabic**: 

```perl
$encoded = encode "buckwalter", decode "arabtex", $decoded
$encoded = encode("buckwalter", decode("arabtex", $decoded))
```

Implemented in **Haskell** and available along with **ElixirFM**: 

```haskell
encoded = encode Buckwalter $ decode ArabTeX decoded
encoded = encode Buckwalter (decode ArabTeX decoded)
encoded = (encode Buckwalter . decode ArabTeX) decoded
```

```cmd
[cmd] decode ArabTeX < decode.d | encode Buckwalter > encode.d
```
Encode Arabic

biruwHi {loIixaA’i ← بُروحُ آلِهاَمِ ← bi-rU.hi al-’i_hA’i

Implemented in Perl and available on CPAN as Encode-Arabic:

$encoded = encode "buckwalter", decode "arabtex", $decoded
$encoded = encode("buckwalter", decode("arabtex", $decoded))

Implemented in Haskell and available along with ElixirFM:

encoded = encode Buckwalter $ decode ArabTeX decoded
encoded = encode Buckwalter (decode ArabTeX decoded)
encoded = (encode Buckwalter . decode ArabTeX) decoded

[cmd] decode ArabTeX < decode.d | encode Buckwalter > encode.d
Encode Arabic

\[
\begin{align*}
\text{biruwHi} & \quad \text{loIixaA’i} \quad \leftrightarrow \quad \text{بُروح} \quad \text{الإِخَاءّ} \quad \leftrightarrow \quad \text{bi-rU.hi} \quad \text{al-’i_hA’i}
\end{align*}
\]

Implemented in **Perl** and available on CPAN as **Encode-Arabic**:

```perl
$encoded = encode "buckwalter", decode "arabtex", $decoded
```

Implemented in **Haskell** and available along with **ElixirFM**:

```haskell
encoded = encode Buckwalter $ decode ArabTeX decoded
```

```haskell
encoded = encode Buckwalter (decode ArabTeX decoded)
```

```haskell
encoded = (encode Buckwalter . decode ArabTeX) decoded
```

```
[cmd] decode ArabTeX < decode.d | encode Buckwalter > encode.d
```
Extensions

Encode Arabic

biruwHi {loIixaA’i ← بروح الآخاء ← bi-rU.hi al-’i_hA’i

Implemented in **Perl** and available on CPAN as **Encode-Arabic**:

```perl
$encoded = encode "buckwalter", decode "arabtex", $decoded
```

Implemented in **Haskell** and available along with **ElixirFM**:

```haskell
encoded = encode Buckwalter $ decode ArabTeX decoded
encoded = encode Buckwalter (decode ArabTeX decoded)
encoded = (encode Buckwalter . decode ArabTeX) decoded
```

```bash
[cmd] decode ArabTeX < decode.d | encode Buckwalter > encode.d
```
biruwHi {loIixaA’i ← بُرَوحَ الآخَاءِ ← bi-rU.hi al-’i_hA’i

Implemented in Perl and available on CPAN as Encode-Arabic:

$encoded = encode "buckwalter", decode "arabtex", $decoded
$encoded = encode("buckwalter", decode("arabtex", $decoded))

Implemented in Haskell and available along with ElixirFM:

encoded = encode Buckwalter $ decode ArabTeX decoded
encoded = encode Buckwalter (decode ArabTeX decoded)
encoded = (encode Buckwalter . decode ArabTeX) decoded

[cmd] decode ArabTeX < decode.d | encode Buckwalter > encode.d
Morphology Disambiguation

Arabic is a language of rich morphology, both derivational and inflectional, with highly ambiguous orthography.
Morphology Disambiguation

Arabic is a language of rich morphology, both derivational and inflectional, with highly ambiguous orthography.

Boundaries of syntactic units, tokens, are obscure in writing—orthographical words, strings, consist of up to four lexemes.
Morphology Disambiguation

Arabic is a language of rich morphology, both derivational and inflectional, with highly ambiguous orthography.

Boundaries of syntactic units, tokens, are obscure in writing—orthographical words, strings, consist of up to four lexemes.

Disambiguation encompasses subproblems like tokenization.
Morphology Disambiguation

Arabic is a language of rich morphology, both derivational and inflectional, with highly ambiguous orthography.

Boundaries of syntactic units, tokens, are obscure in writing—orthographical words, strings, consist of up to four lexemes.

Disambiguation encompasses subproblems like tokenization, full morphological tagging or its simplified ‘part-of-speech’ versions.
Morphology Disambiguation

Arabic is a language of rich morphology, both derivational and inflectional, with highly ambiguous orthography.

Boundaries of syntactic units, tokens, are obscure in writing—orthographical words, strings, consist of up to four lexemes.

Disambiguation encompasses subproblems like tokenization, full morphological tagging or its simplified ‘part-of-speech’ versions, lemmatization
Morphology Disambiguation

Arabic is a language of rich morphology, both derivational and inflectional, with highly ambiguous orthography.

Boundaries of syntactic units, tokens, are obscure in writing—orthographical words, strings, consist of up to four lexemes.

Disambiguation encompasses subproblems like tokenization, full morphological tagging or its simplified ‘part-of-speech’ versions, lemmatization, diacritization or restoration of the structural components of words.
Morphology Disambiguation

Arabic is a language of rich morphology, both derivational and inflectional, with highly ambiguous orthography.

Boundaries of syntactic units, tokens, are obscure in writing—orthographical words, strings, consist of up to four lexemes.

Disambiguation encompasses subproblems like tokenization, full morphological tagging or its simplified ‘part-of-speech’ versions, lemmatization, diacritization or restoration of the structural components of words, plus combinations thereof.
**Linear Lists**

Suppose you can list **morphological analyses** for a given **input string** . . .

<table>
<thead>
<tr>
<th>Morphs</th>
<th>Form</th>
<th>Token Tag</th>
<th>Lemma</th>
<th>Glosses per Morph</th>
</tr>
</thead>
<tbody>
<tr>
<td>lla+(null)</td>
<td>ḍālā</td>
<td>VP-A-3MS--</td>
<td>ḍālā</td>
<td>promise/take an oath + he/it</td>
</tr>
<tr>
<td>lliy~</td>
<td>ḍāliyy</td>
<td>A---------</td>
<td>ḍāliyy</td>
<td>mechanical/automatic</td>
</tr>
<tr>
<td>lliy~+u</td>
<td>ḍāliyy-u</td>
<td>A---------1R</td>
<td>ḍāliyy</td>
<td>mechanical . . . + [def.nom.]</td>
</tr>
<tr>
<td>lliy~+i</td>
<td>ḍāliyy-i</td>
<td>A---------2R</td>
<td>ḍāliyy</td>
<td>mechanical . . . + [def.gen.]</td>
</tr>
<tr>
<td>lliy~+a</td>
<td>ḍāliyy-a</td>
<td>A---------4R</td>
<td>ḍāliyy</td>
<td>mechanical . . . + [def.acc.]</td>
</tr>
<tr>
<td>lliy~+N</td>
<td>ḍāliyy-un</td>
<td>A---------1I</td>
<td>ḍāliyy</td>
<td>mechanical . . . + [indef.nom.]</td>
</tr>
<tr>
<td>lliy~+K</td>
<td>ḍāliyy-in</td>
<td>A---------2I</td>
<td>ḍāliyy</td>
<td>mechanical . . . + [indef.gen.]</td>
</tr>
<tr>
<td>1</td>
<td>ḍāl</td>
<td>N---------R</td>
<td>ḍāl</td>
<td>family/clan +</td>
</tr>
<tr>
<td>+ iy</td>
<td>-ī</td>
<td>S----1-S2-</td>
<td>ū</td>
<td>+ my</td>
</tr>
<tr>
<td>IilaY</td>
<td>ḍīlā</td>
<td>P---------</td>
<td>ḍīlā</td>
<td>to/towards</td>
</tr>
<tr>
<td>Iilay</td>
<td>ḍilay</td>
<td>P---------</td>
<td>ḍīlā</td>
<td>to/towards +</td>
</tr>
<tr>
<td>+ ya</td>
<td>-ya</td>
<td>S----1-S2-</td>
<td>ya</td>
<td>+ me</td>
</tr>
<tr>
<td>0a+liy+(null)</td>
<td>ḍa-li</td>
<td>VIUA-1-S--</td>
<td>waliya</td>
<td>l + follow/come after + [ind.]</td>
</tr>
<tr>
<td>0a+liy+a</td>
<td>ḍa-liyy-a</td>
<td>VISA-1-S--</td>
<td>waliya</td>
<td>l + follow/come after + [sub.]</td>
</tr>
</tbody>
</table>
MorphoTrees

... organize the analyses into a hierarchy with the string as its root
MorphoTrees

... organize the analyses into a hierarchy with the **string** as its **root** and the **full tokens** as the **leaves**
MorphoTrees

... organize the analyses into a hierarchy with the string as its root and the full tokens as the leaves, grouped by their lemmas.
Extensions

MorphoTrees

... organize the analyses into a hierarchy with the string as its root and the full tokens as the leaves, grouped by their lemmas, canonical forms.
MorphoTrees

... organize the analyses into a hierarchy with the string as its root and the full tokens as the leaves, grouped by their lemmas, canonical forms and partitionings of the string into such forms:
Multi-Modal Annotation

and, so to be ready, intend
concern, interest
to understand

understanding

Otakar Smrž (Charles University) Functional Arabic Morphology November 6, 2006 24 / 26
Outline

1 Introduction
2 Morphological Theory
   • Incremental vs. Realizational
   • Lexical vs. Inferential
   • Functional Arabic Morphology
3 Implementation Design
   • ElixirFM
   • Paradigms, parameters, ...
   • Elixir Lexicon
   • FM Generic
4 Extensions
   • Encode Arabic
   • MorphoTrees
5 References
References


