

Programming the Arabic Treebank

Otakar Smrž

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

Dublin City University

April 18, 2008

1 Methodology

- Functional Morphology
- Dependency Syntax
- Tectogrammatics

1 Methodology

- Functional Morphology
- Dependency Syntax
- Tectogrammatics

2 Software

- TrEd Environment
- ElixirFM
- Encode Arabic

- 1 Methodology
 - Functional Morphology
 - Dependency Syntax
 - Tectogrammatics
- 2 Software
 - TrEd Environment
 - ElixirFM
 - Encode Arabic
- 3 References

Prague Arabic Dependency Treebank

PADT is a project of linguistic annotation of **Modern Written Arabic** based on the theory of **Functional Generative Description**.

Prague Arabic Dependency Treebank

PADT is a project of linguistic annotation of **Modern Written Arabic** based on the theory of **Functional Generative Description**.

PADT consists mainly of the **morphological** and **analytical** levels of description. The annotation of **tectogrammatics** and **information structure** is being established.

Prague Arabic Dependency Treebank

PADT is a project of linguistic annotation of **Modern Written Arabic** based on the theory of **Functional Generative Description**.

PADT consists mainly of the **morphological** and **analytical** levels of description. The annotation of **tectogrammatics** and **information structure** is being established.

PADT 1.0 was published in 2004 and has been used by tens of academic and commercial institutions.

Prague Arabic Dependency Treebank

PADT is a project of linguistic annotation of **Modern Written Arabic** based on the theory of **Functional Generative Description**.

PADT consists mainly of the **morphological** and **analytical** levels of description. The annotation of **tectogrammatics** and **information structure** is being established.

PADT 1.0 was published in 2004 and has been used by tens of academic and commercial institutions.

PADT 2.0 is due in 2008 and will cover **over one million** words of text. It merges original **Prague Arabic Dependency Treebank** annotations with converted and enhanced **Penn Arabic Treebank**.

Expected PADT 2.0

Data Set		Corpus 'words'	Functional Morphology			Dependency Syntax			Tectogrammatics		
			tokens	paras	docs	tokens	paras	docs	tokens	paras	docs
Prague	AEP	99360	116717	3006	327	116717	3006	327	9690	242	29
	EAT	48371	55097	1667	207	55097	1667	207	13934	436	58
	ASB	16815	20145	663	44	6527	273	17			
	NHR	21445	25329	426	34	12613	209	17			
	HYT	85683	100537	1782	204	41855	796	91	5228	106	10
	XIN	61500	71548	2389	321	41716	1429	196	2042	75	13
Penn	1v3	141515	161217	4790	628	161217	4790	628			
	2v2	140821	163973	2929	476	163973	2929	476			
	3v2	335250	394466	12445	589	394466	12445	589			
	4v1	161665	192976	6176	397						
Prague		333174	389373	9933	1137	274525	7380	855	30894	859	110
Penn		779251	912632	26340	2090	719656	20164	1693			
PADT 2.0		1112425	1302005	36273	3227	994181	27544	2548	30894	859	110

Expected PADT 2.0

Data Set		Corpus 'words'	Functional Morphology			Dependency Syntax			Tectogrammatics		
			tokens	paras	docs	tokens	paras	docs	tokens	paras	docs
Prague	AEP	99360	116717	3006	327	116717	3006	327	9690	242	29
	EAT	48371	55097	1667	207	55097	1667	207	13934	436	58
	ASB	16815	20145	663	44	6527	273	17			
	NHR	21445	25329	426	34	12613	209	17			
	HYT	85683	100537	1782	204	41855	796	91	5228	106	10
	XIN	61500	71548	2389	321	41716	1429	196	2042	75	13
Penn	1v3	141515	161217	4790	628	161217	4790	628			
	2v2	140821	163973	2929	476	163973	2929	476			
	3v2	335250	394466	12445	589	394466	12445	589			
	4v1	161665	192976	6176	397						
Prague		333174	389373	9933	1137	274525	7380	855	30894	859	110
Penn		779251	912632	26340	2090	719656	20164	1693			
PADT 2.0		1112425	1302005	36273	3227	994181	27544	2548	30894	859	110

The numbers can develop until the official release.

- 1 Methodology
 - Functional Morphology
 - Dependency Syntax
 - Tectogrammatics
- 2 Software
 - TrEd Environment
 - ElixirFM
 - Encode Arabic
- 3 References

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, the **tokens**, are obscure in writing—orthographical words, the **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, the **tokens**, are obscure in writing—orthographical words, the **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, the **tokens**, are obscure in writing—orthographical words, the **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, the **tokens**, are obscure in writing—orthographical words, the **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, the **tokens**, are obscure in writing—orthographical words, the **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, the **tokens**, are obscure in writing—orthographical words, the **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.

He will notify them about that through text messages...

سيخبرهم بذلك عن طريق الرسائل القصيرة...

He will notify them about that through text messages...

... سَيُخَبِّرُهُمْ بِذَلِكَ عَنْ طَرِيقِ الرَّسَائِلِ الْقَصِيرَةِ

String	Token	Tag	Buckwalter Morph Tags	Token Form	Token Gloss
	·	F-----	FUT	sa-	will
سيخبرهم	·	VIIA-3MS--	IV3MS+IV+IVSUFF_MOOD:I	yu-ḥbir-u	he-notify
	·	S----3MP4-	IVSUFF_DO:3MP	-hum	them
بذلك	·	P-----	PREP	bi-	about/by
	·	SD----MS--	DEM_PRON_MS	dālīka	that
عن	·	P-----	PREP	ʿan	by/about
طريق	·	N-----2R	NOUN+CASE_DEF_GEN	ṭarīq-i	way-of
الرسائل	·	N-----2D	DET+NOUN+CASE_DEF_GEN	ar-rasā'il-i	the-letters
القصيرة	·	A-----FS2D	DET+ADJ+NSUFF_FEM_SG+ +CASE_DEF_GEN	al-qaṣīr-at-i	the-short

He will notify them about that through text messages...

... سَيُخَبِّرُهُمْ بِذَلِكَ عَنْ طَرِيقِ الرَّسَائِلِ الْقَصِيرَةِ

String	Token Tag	Buckwalter Morph Tags	Token Form	Token Gloss
	F-----	FUT	sa-	will
سيخبرهم	VIIA-3MS--	IV3MS+IV+IVSUFF_MOOD:I	yu-ḥbir-u	he-notify
	S----3MP4-	IVSUFF_DO:3MP	-hum	them
بذلك	P-----	PREP	bi-	about/by
	SD----MS2-	DEM_PRON_MS	dālīka	that
عن	P-----	PREP	ʿan	by/about
طريق	N-----MS2R	NOUN+CASE_DEF_GEN	ṭarīq-i	way-of
الرسائل	N-----FP2D	DET+NOUN+CASE_DEF_GEN	ar-rasā'il-i	the-letters
القصيرة	A-----FS2D	DET+ADJ+NSUFF_FEM_SG+ +CASE_DEF_GEN	al-qaṣīr-at-i	the-short

Functional Arabic Morphology

Many computational models of Arabic morphology are **lexical** in nature. As they are not designed in connection with any **syntax–morphology interface**, their interpretation is simply **incremental**.

Functional Arabic Morphology

Many computational models of Arabic morphology are **lexical** in nature. As they are not designed in connection with any **syntax–morphology interface**, their interpretation is simply **incremental**.

Functional Arabic Morphology endorses **inferential–realizational** views. It re-establishes the **system** of **inflectional** and **inherent** morphosyntactic properties. It distinguishes various **senses** in which the **properties** are referred to in the grammar.

Functional Arabic Morphology

Many computational models of Arabic morphology are **lexical** in nature. As they are not designed in connection with any **syntax–morphology interface**, their interpretation is simply **incremental**.

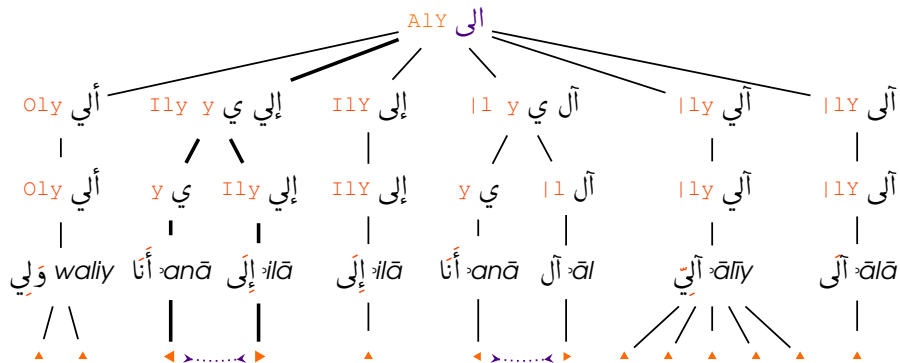
Functional Arabic Morphology endorses **inferential–realizational** views. It re-establishes the **system** of **inflectional** and **inherent** morphosyntactic properties. It distinguishes various **senses** in which the **properties** are referred to in the grammar.

Definition of **lexemes** includes the derivational **root and pattern** information if appropriate. Modeling of the **written** language as well as **spoken** dialects is expected methodologically **identical**.

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

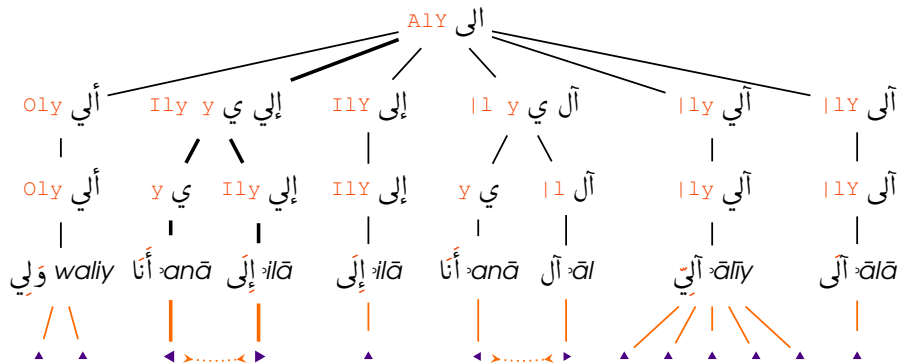
MorphoTrees

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



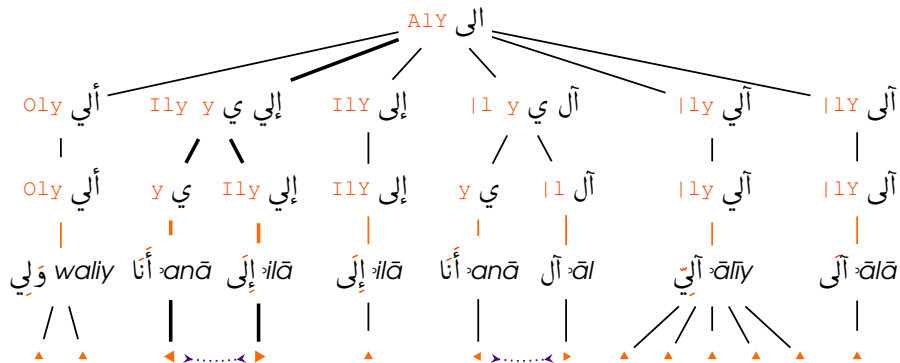
MorphoTrees

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



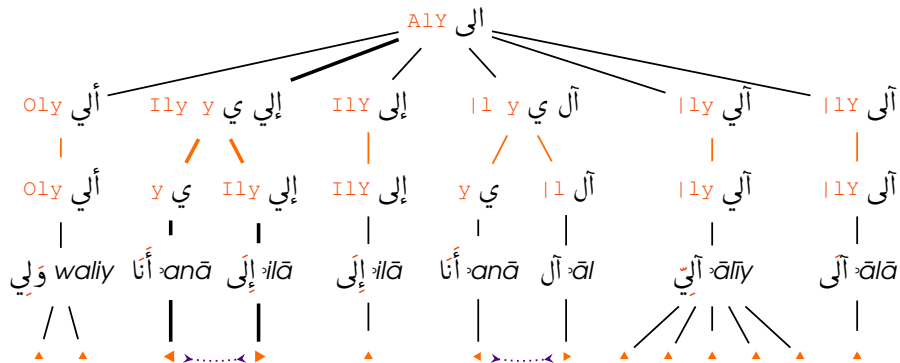
MorphoTrees

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



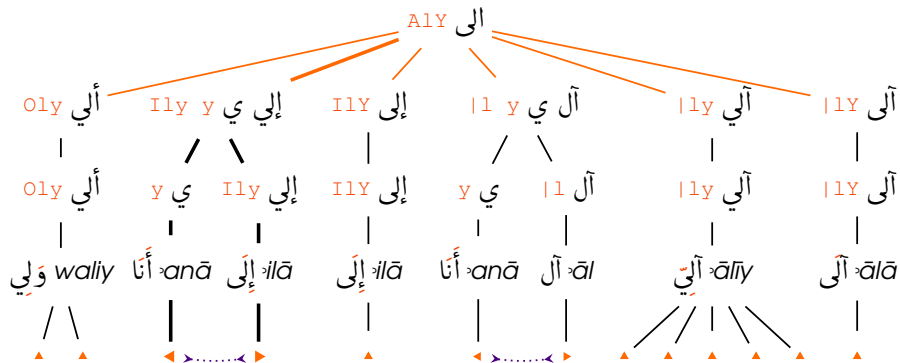
MorphoTrees

... 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**

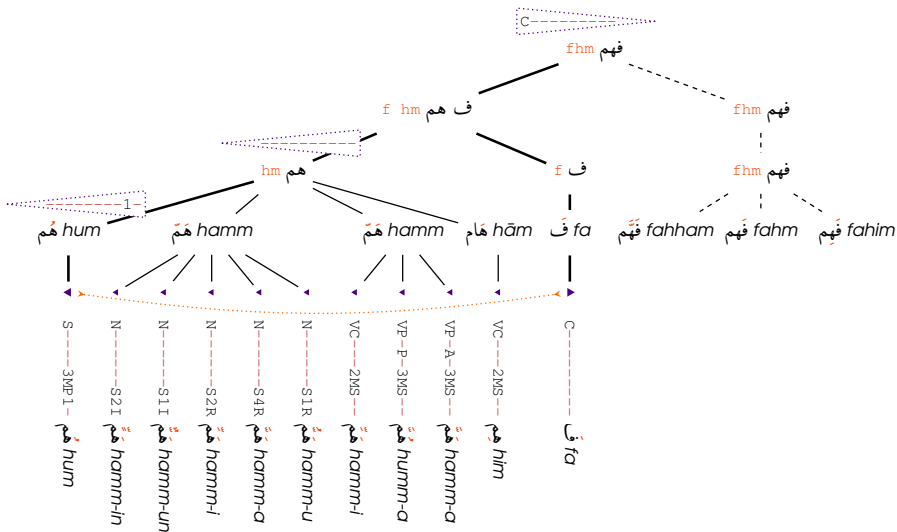


MorphoTrees

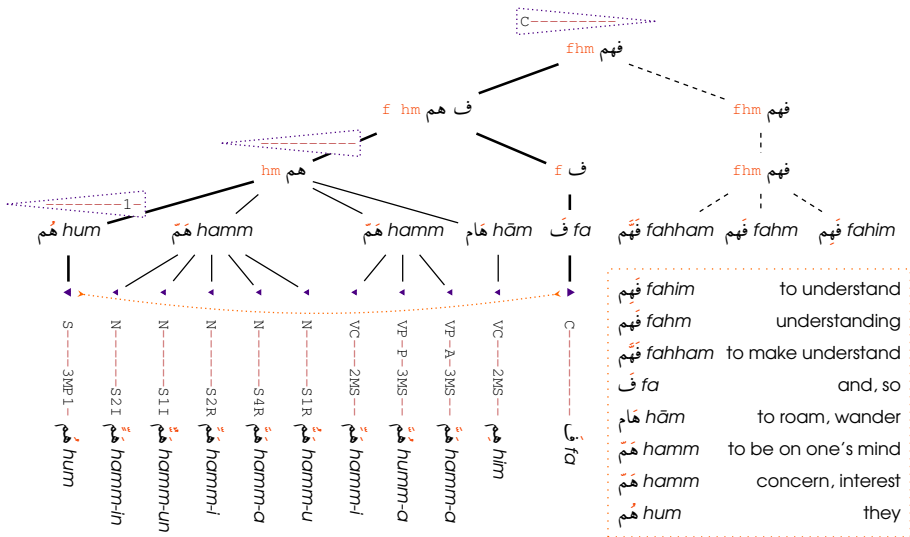
... organize the analyses into a **hierarchy** with the **string** as its **root** and the **full tokens** as the **leaves**, grouped by their **lemmas**, **canonical forms**, and **partitionings** of the string into such forms:



Multi-Modal Annotation



Multi-Modal Annotation



Dependency Syntax

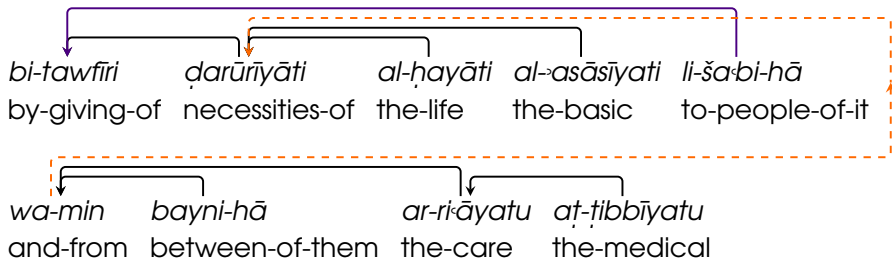
... by providing the basic necessities of life to its people, including medical care

... بِتَوْفِيرِ ضَرُورِيَّاتِ الْحَيَاةِ الْأَسَاسِيَّةِ لِشَعْبِهَا وَمِنْ بَيْنِهَا الرِّعَايَةُ الطَّبِيَّةُ

Dependency Syntax

... by providing the basic necessities of life to its people, including medical care

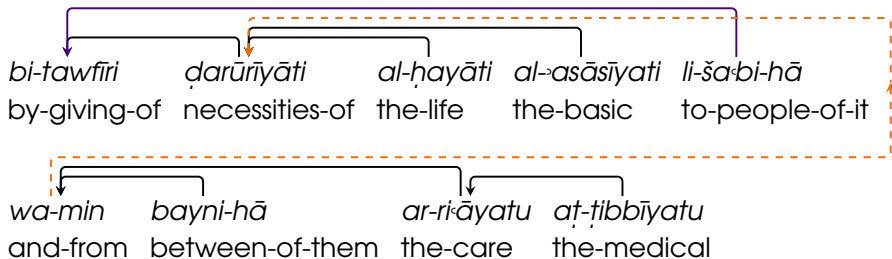
... بِتَوْفِيرِ ضَرُورِيَّاتِ الْحَيَاةِ الْأَسَاسِيَّةِ لِشَعْبِهَا وَمِنْ بَيْنِهَا الرِّعَايَةُ الطَّبِيبَةُ



Dependency Syntax

... by providing the basic necessities of life to its people, including medical care

... بِتَوْفِيرِ ضَرُورِيَّاتِ الْحَيَاةِ الْأَسَاسِيَّةِ لِشَعْبِهَا وَمِنْ بَيْنِهَا الرِّعَايَةُ الطَّبِيبَةُ



Dependency trees capture **distinct types** of relations—**immediate dominance**, **linear precedence**, and **coordination membership**.

In the section on literature, the magazine presented the issue of the Arabic language and the dangers that threaten it.

وفي ملف الأدب طرحت المجلة قضية اللغة العربية والأخطار التي تهددها.



In the section on literature, the magazine presented the issue of the Arabic language and the dangers that threaten it.

وَفِي مَلَفِّ الْأَدَبِ طَرَحَتِ الْمَجَلَّةُ قَضِيَّةَ اللُّغَةِ الْعَرَبِيَّةِ وَالْأَخْطَارِ الَّتِي تَهْدِدُهَا.



Tectogrammatics

Describes **linguistic meaning** in its **semantic** and **pragmatic** aspects. Restores **deep syntax** and marks **information structure**.



Tectogrammatics

Describes **linguistic meaning** in its **semantic** and **pragmatic** aspects. Restores **deep syntax** and marks **information structure**.



- 1 Methodology
 - Functional Morphology
 - Dependency Syntax
 - Tectogrammatics
- 2 Software
 - TrEd Environment
 - ElixirFM
 - Encode Arabic
- 3 References

TrEd Environment

The tree editor **TrEd** is designed and implemented by Petr Pajas. Its distributed version `ntred` is co-authored by Zdeněk Žabokrtský.

<http://ufal.mff.cuni.cz/~pajas/tred/>

TrEd Environment

The tree editor **TrEd** is designed and implemented by Petr Pajas. Its distributed version **ntred** is co-authored by Zdeněk Žabokrtský.

<http://ufal.mff.cuni.cz/~pajas/tred/>

Numerous **annotation contexts** and **macro-style** extensions have been contributed by many other project developers.

The tree editor **TrEd** is designed and implemented by Petr Pajas. Its distributed version **ntred** is co-authored by Zdeněk Žabokrtský.

<http://ufal.mff.cuni.cz/~pajas/tred/>

Numerous **annotation contexts** and **macro-style** extensions have been contributed by many other project developers.

Examples of our work include the **MorphoTrees** context, the **Con-Dep** conversion templates, or miscellaneous scripts for **error detection** and **consistency checking**.

```
'SBAR' => [ [
  [ ["P-----", ".+"], ["C-----", "Oan~a"], undef ],
  [ ["P-----", "baEoda/qabola"], ["C-----", "Oan"], undef ],

  sub { my ($root, undef, @child) = @_;

    @child = map { ConDep($_) } @child;

    PasteNode($child[1], $child[0]);

    PasteNode($_, $child[1]) foreach @child[2 .. @child - 1];

    $child[0]->{'afun'} = "AuxP";
    $child[1]->{'afun'} = "AuxC";

    return $child[0] }

], ... ]
```

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

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

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

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

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

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

Lexicon's Design

The lexicon is stored in a **domain-specific** embedded language.

Lexicon's Design

The lexicon is stored in a **domain-specific embedded** language.

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

Lexicon's Design

The lexicon is stored in a **domain-specific embedded** language.

- (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 it can possibly be **divided** into separate units, as well as be **interlinked** with external **modules**, **without** any **duplication** of information

Lexicon's Design

The lexicon is stored in a **domain-specific embedded** language.

- (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 it can possibly be **divided** into separate units, as well as be **interlinked** with external **modules**, **without** any **duplication** of information
- (c) definition of such a **format** of the lexicon so that **editing** and **understanding** the data is not inappropriately difficult, and using such data **markup** whose syntax is **lightweight** and can be verified with **automatic tools**

```

|> "s l k" <| [
  FaCaL          `verb`   [ "proceed", "behave" ]
    `imperf`    FCuL,
  FiCL          `noun`   [ "wire", "thread" ]
    `plural`    HaFCAL,
  FiCL |< Iy     `adj`   [ "wire", "by wire" ],
  lA >| FiCL |< Iy `adj` [ "wireless", "radio" ],
  FuCuL         `noun` [ "behavior", "conduct" ],
  FuCuL |< Iy   `adj` [ "behavioral" ],
  MaFCaL        `noun` [ "road", "method" ]
    `plural`    MaFACiL ]

```

```

|> "s l k" <| [
  FaCaL          `verb`   [ "proceed", "behave" ]
    `imperf`    FCuL,
  FiCL          `noun`   [ "wire", "thread" ]
    `plural`    HaFCAL,
  FiCL |< Iy     `adj`   [ "wire", "by wire" ],
  lA >| FiCL |< Iy `adj` [ "wireless", "radio" ],
  FuCUL         `noun`   [ "behavior", "conduct" ],
  FuCUL |< Iy   `adj`   [ "behavioral" ],
  MaFCaL        `noun`   [ "road", "method" ]
    `plural`    MaFACiL ]

```

proceed, behave l(u) *salak* سَلَكَ

wire, thread (>aslāk أسلاك) *silk* سِلْك

wire, by wire *silkīy* سِلْكِي

wireless, radio *lā-silkīy* لَاسِلْكِي

behavior, conduct *sulūk* سُلُوك

behavioral *sulūkīy* سُلُوكِي

road, method *maslak* مَسَلِك

(*masālik* مَسَالِك)

ElixirFM implements various utility functions for lookup in the lexicon

ElixirFM implements various utility functions for lookup in the lexicon, inflection and derivation of lexemes

ElixirFM implements various utility functions for lookup in the lexicon, inflection and derivation of lexemes, resolution of strings

ElixirFM implements various utility functions for lookup in the lexicon, inflection and derivation of lexemes, resolution of strings, exporting and pretty-printing of the information, etc.

ElixirFM implements various utility functions for lookup in the lexicon, inflection and derivation of lexemes, resolution of strings, exporting and pretty-printing of the information, etc.

```
lookupEntry "lA-silkIy" ...      lookupReflex "wireless" ...
```

ElixirFM implements various utility functions for lookup in the lexicon, inflection and derivation of lexemes, resolution of strings, exporting and pretty-printing of the information, etc.

```
lookupEntry "lA-silkIy" ...      lookupReflex "wireless" ...
```

```
inflect (lA >| FiCL |< Iy `adj` []) "-----F[SP]-D"
```

ElixirFM implements various utility functions for lookup in the lexicon, inflection and derivation of lexemes, resolution of strings, exporting and pretty-printing of the information, etc.

```
lookupEntry "lA-silkIy" ...      lookupReflex "wireless" ...
```

```
inflect (lA >| FiCL |< Iy `adj` []) "-----F[SP]-D"
```

```
derive ("w .s y" <-> HaFCY `verb` ["recommend"]) "A--P"
```

ElixirFM implements various utility functions for lookup in the lexicon, inflection and derivation of lexemes, resolution of strings, exporting and pretty-printing of the information, etc.

```
lookupEntry "lA-silkIy" ...      lookupReflex "wireless" ...  
  
  inflect (lA >| FiCL |< Iy `adj` []) "-----F[SP]-D"  
  derive ("w .s y" <-> HaFCY `verb` ["recommend"]) "A--P"  
  
resolve "mU.saNY"      resolveBy (omitting "١٥٠" "١٥٠") "موصى"
```

ElixirFM implements various utility functions for lookup in the lexicon, inflection and derivation of lexemes, resolution of strings, exporting and pretty-printing of the information, etc.

```
lookupEntry "lA-silkIy" ...      lookupReflex "wireless" ...  
  
  inflect (lA >| FiCL |< Iy `adj` []) "-----F[SP]-D"  
  derive ("w .s y" <-> HaFCY `verb` ["recommend"]) "A--P"  
  
resolve "mU.saNY"      resolveBy (omitting "موسى" "موصى"  
                                     = -  
                                     = -  
                                     = -) "موصى"  
  
"s l k"      `merge`      al >| lA >| FiCL |< Iy |<< "u"
```

ElixirFM implements various utility functions for lookup in the lexicon, inflection and derivation of lexemes, resolution of strings, exporting and pretty-printing of the information, etc.

```
lookupEntry "lA-silkIy" ... lookupReflex "wireless" ...  
  
inflect (lA >| FiCL |< Iy `adj` []) "-----F[SP]-D"  
derive ("w .s y" <-> HaFCY `verb` ["recommend"]) "A--P"  
  
resolve "mU.saNY" resolveBy (omitting "موسى" "موصى")  
  
"s l k" `merge` al >| lA >| FiCL |< Iy |<< "u"  
"al-lA-silkIyu" al-lā-silkīyu اللاسلكي اللّاسلكي اللّاسلكي
```

Form	All~Asilokiy~apu	اللاسلكية
Morph	Al + lAsilokiy~ + ap + u	
Tag	DET+ADJ+NSUFF_FEM_SG+CASE_DEF_NOM	
Gloss	the + wireless / radio + (fem.sg.) + (def.nom.)	
Lemma	lAsilokiy~_1	لاسلكي

Form	All~Asilokiy~apu	اللاسلكية
Morph	Al + lAsilokiy~ + ap + u	
Tag	DET+ADJ+NSUFF_FEM_SG+CASE_DEF_NOM	
Gloss	the + wireless / radio + (fem.sg.) + (def.nom.)	
Lemma	lAsilokiy~_1	لاسلكي

Form	<i>al-lA-silkIyaTu</i>	<i>al-lā-silkīyatu</i> اللاسلكية
Morph	al > lA > FiCL < Iy < aT << "u"	
Tag	A-----FS1D	
Form	<i>lA-silkIy</i>	<i>lā-silkīy</i> لاسلكي
Morph	lA > FiCL < Iy	
Root	"s l k"	
Reflex	wireless, radio	
Class	adjective	

Form	waOuxoraY	وَأُخْرَى
Morph	wa + OuxoraY	
Tag	CONJ+ADJ	
Gloss	and + other / another / additional	
Lemma	OuxoraY_1	أُخْرَى

Form	waOuxoraY	وَأُخْرَى
Morph	wa + OuxoraY	
Tag	CONJ+ADJ	
Gloss	and + other / another / additional	
Lemma	OuxoraY_1	أُخْرَى

Form	'u_hrY	أُخْرَى	wa	wa
Morph	FuCLY	<< "u"	"wa"	
Tag	A-----FS1I		C-----	
Form	'A_har	آخِر	wa	wa
Morph	HACaL		"wa"	
Root	"' _h r"		"w"	
Reflex	other, another		and	
Class	adjective		conjunction	

Form	sayad~aEiy	سَيَدِّي
Morph	sa + ya + d~aEiy + (null)	
Tag	FUT+IV3MS+IV+IVSUFF_MOOD:I	
Gloss	will + he / it + allege / claim / testify + (ind.)	
Lemma	Aid~aEaY_1	أَدَّي

Form	sayad~aEiy	سَيَدِّي
Morph	sa + ya + d~aEiy + (null)	
Tag	FUT+IV3MS+IV+IVSUFF_MOOD:I	
Gloss	will + he / it + allege / claim / testify + (ind.)	
Lemma	Aid~aEaY_1	ادَّعَى

Form	yadda 'I	yaddaā يَدِّي	sa	sa س
Morph	"ya" >> FtāCI << "u"		"sa"	
Tag	VIIA-3MS--		F-----	
Form	idda 'Y	iddaā ادَّعَى	sa	sa س
Morph	IFtāCY		"sa"	
Root	"d ' w"		"s"	
Reflex	allege, claim, testify		future marker	
Class	verb		particle	

ElixirFM carefully designs the **morphophonemic patterns** of the **templates**, along with the **phonological rules** hidden in the **>|** or **|<<** operators. This greatly simplifies the **morphological rules** proper, inflectional or derivational. ElixirFM implements many **generalizations** of classical grammars, and suggest some new ones.

"ya" >>| FtaCI |<< "u"

yadda `I

yadda`ā يَدْعِي

"ya" >>| FtaCI |<< "a"

yadda `iya

yadda`iya يَدْعِي

"ya" >>| FtaCI |<< ""

yadda `i

yadda`i يَدْعِ

ElixirFM carefully designs the **morphophonemic patterns** of the **templates**, along with the **phonological rules** hidden in the **>|** or **|<<** operators. This greatly simplifies the **morphological rules** proper, inflectional or derivational. ElixirFM implements many **generalizations** of classical grammars, and suggest some new ones.

"ya" >> FtaCI << "u"	yadda `I	yaddaī يَدَّعِي
"ya" >> FtaCI << "a"	yadda `iya	yaddaīya يَدَّعِي
"ya" >> FtaCI << ""	yadda `i	yaddaī يدَّع
"ya" >> FCuL << "u"	yaktubu	yaktubu يَكْتُبُ
"ya" >> FCuL << "a"	yaktuba	yaktuba يَكْتُبُ
"ya" >> FCuL << ""	yaktub	yaktub يَكْتُبُ

Buckwalter Transliteration

يُولَدُ جَمِيعُ النَّاسِ أحرارًا مُتساوِينَ في الكَرَامَةِ وَالْحَقوقِ. وَقَد وَهَبوا عَقْلاً وَضَميراً وَعَليهِم
أَنْ يُعَامِلَ بَعْضُهُم بَعْضاً بِرُوحِ الإِخاءِ.

```
yuwladu jamiyEu {ln~aAsi OaHoraArFA mutasaAwiyna fiy  
{lokaraAmapi wa{loHuquwqi. waqado wuhibuwa EaQolAF  
waDamiyrFA waEalayohimo Oano yuEaAmila baEoDuhumo baEoDFA  
biruwHi {loIixaA'i.
```

يولد جميع الناس أحرارا متساوين في الكرامة والحقوق. وقد وهبوا عقلا وضميرا وعليهم
أن يعامل بعضهم بعضا بروح الإخاء.

```
ywld jmyE AlnAs OHRArA mtsAwyn fy AlkrAmp wAlHqwq. wqd  
whbwA EqLA wDmyrA wElyhm On yEAml bEDhm bEDA brwH AlIXA'.
```

Notation of ArabTeX

يُولَدُ جَمِيعُ النَّاسِ أَحْرَارًا مُتَسَاوِينَ فِي الْكِرَامَةِ وَالْحُقُوقِ. وَقَدْ وَهَبُوا عَقْلًا وَضَمِيرًا وَعَلَيْهِمْ أَنْ يُعَامَلَ بَعْضُهُمْ بَعْضًا بِرُوحِ الْإِخَاءِ.

يولد جميع الناس أحرارا متساوين في الكرامة والحقوق. وقد وهبوا عقلا وضميرا وعليهم أن يعامل بعضهم بعضا بروح الإخاء.

*Yūladu ġamīu 'n-nāsi aħrāran mutasāwīna fī 'l-karāmati wa-'l-ħuqūqi.
Wa-qad wuhibū aqlan wa-ḍamīran wa-alay-him an yuāmila baḍu-
hum baḍan bi-rūħi 'l-iħā'i.*

```
\cap yUladu ^gamI`u an-nAsi `a.hrAraN mutasAwIna fI  
al-karAmATi wa-al-.huqUqi.
```

```
\cap wa-qad wuhibUW `aqlaN wa-.damIran wa-`alay-him `an  
yu`Amila ba`.du-hum ba`.daN bi-rU.hi al-`i_hA`i.
```

Encode Arabic

biruwHi {loIixaA'i بِرُوحِ الْإِخَاءِ *bi-rūḥi 'l-iḥā'i* *bi-rU.hi al-'i_hA'i*

Encode Arabic

biruwHi {loIixaA'i بِرُوحِ الْإِخَاءِ *bi-rūḥi 'l-iḥā'i* *bi-rU.hi al-'i_hA'i*

[>] decode ArabTeX < decode.d | encode Buckwalter > encode.d

Encode Arabic

biruwHi {loIixaA'i بِرُوحِ الْإِخَاءِ *bi-rūḥi 'l-iḥā'i* *bi-rU.hi al-'i_hA'i*

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

Implemented in [Perl](#) and available on CPAN as [Encode-Arabic](#):

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

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

Encode Arabic

biruwHi {loIixaA'i بِرُوحِ الْإِخَاءِ *bi-rūḥi 'l-iḥā'i bi-rU.hi al-'i_hA'i*

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

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
```

References

ElixirFM plus lexicons, Encode Arabic, MorphoTrees, and Arab \TeX extensions are open-source software licensed under GNU GPL:

<http://sourceforge.net/projects/elixir-fm/>

<http://sourceforge.net/projects/encode-arabic/>

References

ElixirFM plus lexicons, Encode Arabic, MorphoTrees, and Arab \TeX extensions are open-source software licensed under GNU GPL:

<http://sourceforge.net/projects/elixir-fm/>

<http://sourceforge.net/projects/encode-arabic/>

Prague Arabic Dependency Treebank ++ is the project's weblog:

<http://ufal.mff.cuni.cz/padt/online/>

- Buckwalter, Tim. **Buckwalter Arabic Morphological Analyzer 1.0**. LDC2002L49, ISBN 1-58563-257-0. 2002
- Forsberg, Markus and Aarne Ranta. **Functional Morphology**. Proceedings of ICFP 2004, pages 213–223. ACM Press. 2004
- Lagally, Klaus. **ArabTeX: Typesetting Arabic and Hebrew, User Manual Version 4.00**. Technical Report 2004/03, Fakultät Informatik, Universität Stuttgart. 2004
- Sgall, Petr and Eva Hajičová and Jarmila Panevová. **The Meaning of the Sentence in Its Semantic and Pragmatic Aspects**. Academia, Prague. 1986

- Smrž, Otakar and Petr Pajas. **MorphoTrees of Arabic and Their Annotation in the TrEd Environment**. Proceedings of the NEMLAR Conference 2004, pages 38–41. 2004
- Smrž, Otakar. **Functional Arabic Morphology. Formal System and Implementation**. Ph.D. thesis, Charles University in Prague. 2007
- Smrž, Otakar et al. **Prague Arabic Dependency Treebank: A Word on the Million Words**. LREC 2008 Workshop on Arabic and Local Languages. 2008