ArabTAG: from a Handcrafted to a Semi-automatically Generated TAG

Chérifa Ben Khelil[1,2] Denys Duchier[1] Yannick Parmentier[1] Chiraz Zribi[2] Fériel Ben Fraj[2]
(1) LIFO - Université d'Orléans, France
firstname.lastname@univ-orleans.fr
(2) RIADI - ENSI, Université La Manouba, Tunisia

Abstract

In this paper, we present the redesign of an existing TAG for Arabic using a description language (so-called metagrammatical language). The use of such a language makes it easier for the linguist to share information among grammatical structures while ensuring a high degree of modularity within the target grammar. Additionally, this redesign benefits from a grammar testing environment which is used to check both grammar coverage and overgeneration.

1 Introduction

Precision grammars provide fine-grained descriptions of languages, which are beneficial for many NLP applications such as Dialog systems, Question-Answering systems, Automatic Summarization systems, etc. Still the development of such language resources is limited for it comes at a high cost. For instance, the development of the first large-coverage TAG for French took more than 10 person-years (Abeillé et al., 1999). In this context, many efforts have been put into semi-automatic grammar production, either in a data-oriented fashion (that is, by acquiring grammar rules from annotated corpora), or in a knowledge-based fashion (that is, by using description languages to capture generalizations among grammar rules).

A particularly interesting feature of the latter is that it offers a relatively good control on the grammar being produced. This allows among others to extend the produced grammar with various levels of description such as morphology or semantics (see e.g. (Duchier et al., 2012) or (Gardent, 2008)).

In this paper, we are interested in applying such grammar production techniques to the description of Arabic. Arabic is a challenging language when it comes to grammar production for it exhibits specific features such as (i) a relatively free word order, combined with (ii) a rich morphology, and (iii) the omission of diacritics (vowels) in written texts. These features motivates the use of highly-expressive description languages. In this work, we use the XMG description language (Crabbé et al., 2013) to describe in an expressive and yet concise way the syntax of Arabic. This description is based on an existing handcrafted Tree-Adjoining Grammar for Arabic named ArabTAG (Ben Fraj, 2011), which serves as a model for the current development.

The paper is organized as follows. In Section 2, we present the grammatical resource, namely ArabTAG, this work is based on. We particularly comment on the ArabTAG grammar coverage, its design choices and limitations. In Section 3, we present the XMG description language, together with recent advances in its implementation. These advances include a new level of modularity brought by the possibility to dynamically define the description language needed by the linguist in a given context and to *compile the compiler* for this description language.[1] In Section 4, we present a concise description of the syntax of Arabic using XMG. In particular, we show how to deal with the free word order of this language. We also present a testing environment which was designed to facilitate grammar development. In Section 5, we compare our approach with related work. Finally in Section 6, we con-

[1] As such, XMG2 can be seen as a meta interpreter which takes as an input a compiler specification and produces as an output a compiler for a description language. The latter is used by linguists to describe grammatical resources, as explained later in this paper.

Proceedings of the 12th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+12), pages 18–26, Düsseldorf, Germany, June 29 - July 1, 2016.

clude and present some short-term perspectives of development of the grammar (including morphological and semantic information).

2 ArabTAG: a Tree-Adjoining Grammar for Arabic

As mentioned above, Arabic language combines complex linguistic phenomena that make its processing particularly ambiguous. Let us recall briefly what these are:

- The diversity of lexical and grammatical interpretation of an Arabic word. This ambiguity is due to the absence of vocalic signs, which is frequent in modern Arabic.

- The order of the sentence's constituents in Arabic language is free. For a verbal sentence composed of **V**erb, **S**ubject and **O**bject, we can have three combinations (VSO, VOS, SVO) that are all syntactically correct.

- The phenomenon of agglutination: in order to have more complex forms, clitics can tie up with words. Thus, a sentence can correspond to just one agglutinative form as in the example فأسقيناكموه (which is the longest word in Quran) :

  (1) ف أسقي نا كم و ه
  it to you we gave-to-drink and
  'and we gave it to you to drink'

  It is composed of a conjunction, verb, subject and two objects which are all enclosed in the same textual form.

- The recursive structures whose length is not limited:[2]

  (2) الذي لم ينقطع طوال اللّيل
  the-night during stop not that
  أيقظته بنباحها
  with-their-barking aroused-that-him
  الكلاب هي الّتي
  those are the-dogs
  'the dogs are those that aroused him with their barking that does not stop during the night'

[2]This example is taken from (Ben Fraj, 2010).

Digital resources (grammars and treebanks) which can be used for parsing Arabic texts are scarce. In this context, let us describe a semi-lexicalized Arabic Tree-Adjoining Grammar called ArabTAG (Ben Fraj, 2010).

ArabTAG has been developed at RIADI Laboratory, National School of Computer Sciences, University of La Manouba, Tunisia. It includes a set of elementary trees representing the basic syntactic structures of Arabic. The construction of these structures was based on school grammar books and books of Arabic grammar such as (Kouloughli, 1992). To enhance interoperability between tools and resources, all these structures were encoded in XML.

ArabTAG has been used to build a treebank for Arabic. More precisely, ArabTAG's elementary trees served as representative structures to annotate syntactically a corpus. The resulting treebank consists of 950 sentences (5000 words) annotated with their corresponding TAG derivation trees.

2.1 ArabTAG and lexicalization

ArabTAG is semi-lexicalized since it contains trees where all nodes are labelled with non-terminal symbols (that is, syntactic categories). More precisely, ArabTAG contains two sets of elementary trees: lexicalized trees (that is, having at least one lexical item as a leaf node) and patterns trees. The choice of the semi-lexicalized TAG variant was made in order to reduce the important number of possible syntactic structures. The lexicalized trees are reserved to prepositions, modifiers, conjunctions, demonstratives, etc. On the other hand, the patterns trees represent verbs, nouns, adjectives or any kind of phrases.[3] These elementary trees are enriched by different information organized in feature structures.

2.2 ArabTAG coverage

To represent Arabic basic structures, ArabTAG contains 24 lexicalized elementary trees and 241 pattern-based elementary trees. Lexicalized trees correspond to the particles as the prepositions, conjunctions, interrogatives, etc.

These trees correspond to the simple syntactic structures of the Arabic sentences (nominal and verbal), all classes of phrasal structures (NP (Nominal Phrase), PP (Prepositional

[3]Concretely, this means that ArabTAG is largely made of tree schemata (unanchored lexicalized TAG trees) like XTAG (XTAG Research Group, 2001).

Phrase)) as well as the different sub-classes of the phrasal structures (Adjectival NP, Complement NP, Propositional NP). Moreover, ArabTAG presents different kinds of sentences: active, passive, interrogative, complex sentences. In addition, it covers elliptical, anaphoric and subordinate structures. It takes into account the change of the order of the sentence's components and the agglutinative forms. The current version of this grammar has some limitations that can be summarized as follows:

- The syntactic structures enriched with supplements (circumstantial complements of time, place, etc.) are not described.
- The representation of forms of agglutination is not well reflected in ArabTAG. These forms should be extended to improve the coverage of the grammar.
- ArabTAG emphasizes syntactic relations without regard to semantic information. However, syntactic interpretation cannot be complete if it does not involve semantic information.
- ArabTAG consists of a flat set of elementary trees (that is, without any structure sharing). In particular, it is not organized in a hierarchical way.

Therefore we propose a new version ArabTAG V2 that takes into account the aspects mentioned above. This new version is being enhanced with a new organization (using a description language to specify elementary trees) and new unification criteria (both at the syntactic and semantic-syntactic levels). Coverage is also being extended by adding elementary trees for the representation of additional complements. Finally, as will be shown in Section 4, we are currently in the process of structuring the grammar into a hierarchical organization.

3 XMG2: a new Generation of Description Languages

As mentioned above, a common strategy to semi-automatically produce precision grammars is to use a description language. Such a language permits the linguist to formally specify the structures of a target grammar. When this language comes with an implementation (that is, a compiler), the grammar specification (so-called *metagrammar*) can be compiled into an actual electronic grammar.

Designing and implementing description languages for grammar specification is a field which has been quite active over the past decades, seminal work includes languages such as PATRII (Shieber et al., 1983), DATR (Evans and Gazdar, 1996), LexOrg (Xia, 2001), DyALog (Villemonte De La Clergerie, 2010), or more recently XMG (Crabbé et al., 2013). These languages differ among others, in the way variables are handled (local versus global scopes) and how structure sharing is represented (inheritance versus transformations). Here, we propose to use XMG to describe Arabic for it exhibits particularly pertinent features :

- it is highly expressive, which makes it possible to define highly factorized grammar descriptions (in our case, this will be used to deal with semi-free word order) ;
- it is particularly adapted to the description of tree grammars (it has been used to develop several electronic TAG grammars for e.g. French, English, German, Vietnamese, Korean) ;
- it is highly extensible (as will be described in this Section, it can be *configured* to describe various levels of language, such as semantics or morphology) ;
- it is open-source and actively developed.

The first version of XMG was based on the following two main concepts:[4]

1. elementary trees are made of common reusable tree fragments, which can be combined conjunctively or disjunctively ;
2. each of these fragments can be (i) specified using a tree description logic such as the one defined by Rogers and Vijay-Shanker (1994) and (ii) encapsulated within *classes*.

On top of these, XMG includes a facility to define other types of reusable information (so-called *dimensions*), which can be used to extend tree fragments with e.g. semantic formulas.

[4] A detailed introduction to XMG can be found in (Crabbé et al., 2013).

An important limitation of XMG was the limited number of dimensions it can handle, namely syntax (tree fragments), semantics (predicate logic formulas) and syntax-semantic interface (attribute-value matrices). In order to make it possible for the linguist to describe several levels of language (whatever their number is), and to combine various data structures (not only tree fragments or predicate logic formulas), XMG was extended[5] as described in the following subsections.

3.1 Describing description languages

In order to allow for the description of an unlimited number of dimensions, XMG should include a support for user-defined dimensions. This involves (i) being able to formally define the description language for this dimension and (ii) being able to interpret formulas of this language to output valid linguistic structures.

To do this, XMG2 extends XMG by including a *meta* metagrammar compiler (Petitjean, 2014).[6] More concretely, the modular architecture of XMG2 makes it possible for contributors to develop so-called *language bricks*. A brick is the formal definition of a description language (that is, the CFG underlying this language) together with the implementation of the interpretation procedure for (abstract syntax trees of) formulas of this language. The meta metagrammar compiler can then compile *on-demand* sets of language bricks, which can then be used together to describe various levels of linguistic structures. In other words, the metagrammar compiler is now compiled from language bricks (hence the arbitrary number of dimensions).

As an illustration, let us consider the following bricks which are used to describe TAG grammars.[7]

Language brick for combining descriptions:

$$\begin{array}{rcl} \mathit{Desc} & ::= & \mathit{Stmt} \\ & | & \mathit{Stmt} \wedge \mathit{Desc} \\ & | & \mathit{Stmt} \vee \mathit{Desc} \end{array}$$

Basically descriptions are made of statements which can be combined either conjunctively or disjunctively.

Language brick for describing tree fragments:

$$\begin{array}{rcl} \mathit{Stmt} & ::= & \texttt{node}\ \mathit{id} \\ & | & \texttt{node}\ \mathit{id}\ \mathit{AVM} \\ & | & \mathit{id} \lhd \mathit{id} \quad | \quad \mathit{id} \lhd^{+} \mathit{id} \\ & | & \mathit{id} \prec \mathit{id} \quad | \quad \mathit{id} \prec^{+} \mathit{id} \end{array}$$

Tree fragments are described using a tree description logic based on dominance (written $\lhd$) and precedence (written $\prec$) relations between node variables (identifiers).[8]

Language brick for describing Attribute-Value Matrices (AVMs):

$$\begin{array}{rcl} \mathit{AVM} & ::= & [\mathit{Feats}] \\ \mathit{Feats} & ::= & \mathit{Feat} \\ & | & \mathit{Feat}, \mathit{Feats} \\ \mathit{Feat} & ::= & \mathit{id} = \mathit{Value} \end{array}$$

AVMs are sets of pairs (written *Feats* here). Each pair associates a feature (i.e. an identifier) with a value.

The interpretation of descriptions based on these bricks requires (i) unification (for interpreting AVMs and node or feature variable unification) and (ii) tree description solving. Hence the contributor has to write the Prolog code which performs these treatments.[9]

3.2 Assembling description languages

From the library of language bricks available within XMG2, metagrammar designers can load the necessary bricks to describe their target formalism. This is done by declaring within a YAML file, which bricks are to be loaded.

For instance, in order to describe TAG grammars using the above-mentioned bricks, one would have to write the following YAML file:

```
mg:
  _Stmt: control
control:
  _Stmt: dim_syn
dim_syn:
  tag: "syn"
  _Stmt: syn
syn:
  _AVM: avm
avm:
  _Expr: feats
feats:
  _Value: value
```

[5] And at the same time re-implemented in Prolog for the previous version of XMG was coded in the Oz Programming Language, which is no longer maintained.

[6] See `https://launchpad.net/xmg-ng`.

[7] For a detailed introduction to the description of TAG grammars with XMG2, see (Petitjean, 2014).

[8] The $^{+}$ refers to the relation's transitive closure.

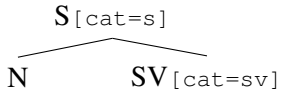
[9] As described in (Petitjean, 2014), the contributor actually has to provide a repository where each compilation step is specified in Prolog.

Such a specification indicates that a metagrammar (`mg`) uses statements of type `control` (that is, which make use of our language brick for combining descriptions). This `control` brick itself uses statements of type `dim_syn`. Formulas of this type are declared using the keyword `"syn"` and contain statements of type `syn`. This brick `syn` itself uses externally defined AVMs. AVMs are made of features whose expressions are values. Again, note that this example is simplified, for instance nodes here are only decorated with AVMs, while in XMG2, they are also decorated with properties such as marks (e.g. $\downarrow$ for substitution) or colors (to guide node identification when solving tree descriptions).

3.3 Using assembled languages to describe natural language

Once the metagrammar language is fully specified and the compiler for this language compiled, the metagrammar designer can write the description of the target linguistic resource. In our case, such a resource is a TAG grammar. It is described as (conjunctive and disjunctive) combinations of tree fragments. Such fragments are defined as formulas of a tree description logic based on dominance and precedence relations between node variables.

For instance the following tree fragment:



would be represented in an XMG2 metagrammatical description as follows:

$$\begin{aligned} &\textbf{node}\ S\ [cat{=}s] \\ \wedge\ &\textbf{node}\ N\ [cat{=}n] \\ \wedge\ &\textbf{node}\ SV\ [cat{=}sv] \\ \wedge\ &s \triangleleft n\ \wedge\ s \triangleleft sv\ \wedge\ n \prec sv \end{aligned}$$

Note that, like in XMG, tree fragments in XMG2 are encapsulated within classes. A class corresponds to the association of a description (i.e., a combination of statements) with a name (making it possible to reuse a given description in various contexts). Unification variables (used to refer either to a node, a feature, or a value) are declared within classes (and their scope is by default limited to the class). When classes are combined, variables denoting the same information need to be explicitly unified (using the operator $=$). While providing the user with flexibility (variable names can be freely used without any risk of conflict), this local scope and explicit unification hampers grammar design as shown by Crabbé *et al.* (2013). So XMG2 comes with an alternative handling of unification which was already present in XMG, namely *node coloring*. Each node of the metagrammatical description can be colored in either black, white or red. A black node is a *resource* and can be unified with 0 or more white nodes; a white node is a *need* and must be unified with a black node; a red node is saturated and cannot be unified with any other node. Since metagrammar compilation solves descriptions in order to compute minimal tree models, variable denoting nodes are implicitly unified due to dominance constraints but also the above-mentioned color constraints. Colors are an elegant way to restrict and guide the ways in which variables can be unified, and so in which tree fragments can be combined.

In the next section, we will see how such a modular and expressive description language is being used to (re)describe the syntax of Arabic.

4 ArabTAG Revisited using XMG2

In this section, we aim to illustrate the flavor of the new formulation of ArabTAG by focusing on the modelization of simple verb subcategorization frames for matrix clauses. In order to make this presentation more easily accessible, we use a combination of logical and graphical notation rather than XMG's concrete syntax.

4.1 Describing verbal predicates in Arabic with XMG2

In an Arabic matrix clause, the verb and its arguments can mostly be freely reordered.[10] Since ArabTAG made the choice of flat trees for verbal constituents, the TAG grammar must supply initial trees for all possible permutations of the arguments. Thanks to the tree description based approach of XMG, this is easily achieved simply by not stipulating any precedence constraint among these arguments.

EpineVerbe(C) is an abstraction that contributes a fragment of tree description for the verbal spine of a matrix clause. Since adverbs can be freely interspersed between arguments, we need to provide appropriate adjunction points for them. AG

[10] Words are usually represented in Arabic in a VSO order, still alternative orders may be used as well with morphological constraints on the verb. See e.g. (El Kassas and Kahane, 2004).

is an adjunction point allowing an adverb at the front of the clause. AD is an adjunction point for inserting an adverb after the verb (or after an argument as we will see later). Nodes here are colored (represented by B, W and R for black, white and red respectively). EpineVerbe is parameterized by a color C as depicted below:[11]

EpineVerbe(C) $\longrightarrow$
  SV^C [cat=sv]
  |
  AG^C [cat=advg]
  |
  AD^C [cat=advd]
  |
  $\text{V}^C_\diamond$ [cat=v]

MatrixClause contributes the actual verb spine (which can be seen as a resource) and therefore instantiates EpineVerbe with the color black:

MatrixClause $\longrightarrow$ EpineVerbe(B)

EpineArg is an abstraction used for attaching to the verbal spine a tree description for an argument (seen as a need).

EpineArg $\longrightarrow$
  [AG] $\Leftarrow$ EpineVerbe(W)
  $\wedge$ AD^{R} [cat=advd] $\wedge$ AG $\triangleleft$ AD

It instantiates EpineVerbe with the color white, thus forcing it to unify with the actual verb spine. [AG] $\Leftarrow$ EpineVerbe(W) additionally imports into the current scope the variable AG provided by EpineVerbe, and attaches a new AD adjunction node for optional insertion of an adverb after the argument. Note that AG $\triangleleft$ AD only specifies that AG immediately dominates AD, but introduces no precedence constraint.

SujetCanon instantiates EpineArg and attaches an SN substitution node below the argument AD supplied by EpineArg:

SujetCanon $\longrightarrow$
  [AD] $\Leftarrow$ EpineArg()
  $\wedge$ $\text{SN}^{\text{R}}_\downarrow$ [cat=sn,cas=nom]
  $\wedge$ AD $\triangleleft$ SN

A direct object normally appears after the verb. If it appears before the verb, then it gives rise to a cleft-construction and requires an object-clitic marker on the verb (boolean feature oclit):

[11] In our metagrammatical description of Arabic syntax, tree fragment names are in French (e.g. EpineVerbe) and so are syntactic categories (e.g. SV for *Syntagme Verbal*).

ObjetCanonSN $\longrightarrow$
  [AD, V] $\Leftarrow$ EpineArg()
  $\wedge$ $\text{SN}^{\text{R}}_\downarrow$ [cat=sn, cas=acc]
  $\wedge$ AD $\triangleleft$ SN
  $\wedge$ ((V [oclit=$-$] $\wedge$ V $\prec^+$ SN) $\vee$
    (V [oclit=$+$] $\wedge$ SN $\prec^+$ V))

The direct object can also be just a clitic:

ObjetCanonClit $\longrightarrow$
  [V] $\Leftarrow$ EpineVerbe(W)
  $\wedge$ V [oclit=+]

ObjetCanon $\longrightarrow$ ObjetCanonSN $\vee$
  ObjetCanonClit

The indirect object requires a particle PV (stipulated by the verb) that is realized either as a separate preposition or as a morphological affix on the noun.

ObjetIndCanon $\longrightarrow$
  [AD, V] $\Leftarrow$ EpineArg()
  $\wedge$ SP^{B} [cat=sp]
    $\text{P}^{\text{R}}_\diamond$ $\text{SN}^{\text{R}}_\downarrow$ [cat=sn, cas=gen]
  $\wedge$ AD $\triangleleft$ SP $\wedge$ V [p=PV]
  $\wedge$ ((P [phon=ε] $\wedge$ SN [p=PV]) $\vee$
    (P [phon=PV] $\wedge$ SN [p=ε]))

Finally the 3 basic verb families can be obtained as follows:

Intransitive $\longrightarrow$ MatrixClause $\wedge$ SujetCanon
Transitive $\longrightarrow$ Intransitive $\wedge$ ObjetCanon
DiTransitive $\longrightarrow$ Transitive $\wedge$ ObjetIndCanon

We saw that using a metagrammatical language based on (i) combinations of reusable tree fragments together with (ii) a tree description logic allowing for expressing (underspecified) dominance and precedence relations between nodes, makes it possible to describe the syntax of verbal predicates in Arabic in a concise and modular way. This metagrammatical description relies on linguistic motivations (e.g., alternative realizations of grammatical functions, valence, etc.), and can be easily extended by just adding missing tree fragments and combination rules (in this sense, the metagrammatical language is monotonic, since no fragment deletion needs to be expressed, only alternatives).

Note that this metagrammatical description language was already available within XMG. The benefit of using XMG2 will come shortly once the metagrammar will be extended with additional information such as morphological descriptions (see *infra*). Indeed, XMG2 will be needed to assemble a metagrammatical language that does not only

permit the linguist to describe syntactic trees but other levels of description (e.g. morphological structures), which could be *connected* with each other (via shared unification variables).

4.2 Current state of ArabTAGv2

As mentioned above, the work presented here is based on an existing TAG for Arabic (ArabTAGv1), which is handcrafted (the linguist uses a specific tool to describe elementary trees, this tool performs additional consistency checks during grammar development and also some predictions on the structures being described). ArabTAGv1 contains 380 elementary trees, and was developed in the context of (Ben Fraj, 2010). 83% out of these 380 trees (that is, 315 trees) represent verbal predicates.

Our work is still in its early stage. The redesign of ArabTAG using a metagrammar started 4 months ago in the context of a PhD co-supervision. So far, we generated 114 trees from a description made of 30 classes (that is, 30 tree fragments or combination rules). The metagrammar is about 600 lines long. We focused on verbal predicates and are now working on nominal phrases. We aim at out-performing the coverage of ArabTAGv1 within a couple of years, while extending it with morphological and semantic information (which impact syntax, e.g. word order or agreements).

Arabic exhibits challenging properties including its rich morphology (making use, among others, of agglutination). We plan to integrate a morphological dimension in our metagrammatical description following seminal work by Duchier *et al.* (2012). The idea is to generate inflected forms from a morphological meta-description. This meta-description uses a two-layer representation. First a constraint-based description of morphological information (represented as ordered and potentially empty fields) is defined. Then, surface transformations (e.g. related to agglutination) are captured by means of postprocessings (in our case rewriting rules). The metagrammar compiler for this morphological meta-description is compiled from the selection of adequate description languages from XMG2's library of language bricks.

4.3 About metagrammar development

While designing ArabTAG with XMG2, we set up a development environment in order to check grammar coverage (in particular aiming at reducing both under and over-generation).[12] More concretely, together with the metagrammar which actually consists of tree templates, we are also designing proof-of-concept syntactic and morphological lexicons for Arabic, following the 3-layer lexicon architecture (tree templates, lemmas, words) of the XTAG project (2001). Each new syntactic phenomena included in ArabTAG leads to the extension of a test corpus gathering both grammatical and ungrammatical sentences (associated with the number of expected parses). The TuLiPA parser (Parmentier et al., 2008) is then run on the test corpus to check the quality of the grammar, producing logs which can help metagrammar designer to fix potential bugs in the metagrammatical description. An extract of these logs is given in Fig. 1, and an example of derived tree in Fig. 2.[13]

```
Axiom:  sv
Anchoring failed on tree
Interrogative_2--متى for lexical item
متى
Grammar anchoring time:  0.023071728
sec.
@@##Tree combinations before classical
polarity filtering :  16
@@##Tree combinations after classical
polarity filtering :  2
Grammar conversion time:  0.051578702
sec.
Parsing time:  0.044386232 sec.
Sentence "إلى متى نامَ عليٌ" parsed.
Forest extraction time:  0.003370246
sec.
Number of derivation trees 1
Parses available (in XML) in
corpus0.xml.
XML production time:  0.418911166 sec.
Total parsing time for sentence
"إلى متى نامَ عليٌ" :  0.541318074 sec.
```

Figure 1: Log file produced during the development of ArabTAG (extract)

5 Related Work

To our knowledge, there are very few TAG-based descriptions of Arabic, the main attempt at such a description being work by Habash and Rambow (2004), where a tree-adjoining grammar was extracted from an Arabic Treebank (namely the Penn

[12] This development environment consists of Python scripts.

[13] Note that the sentence is displayed in the discourse direction (e.g., from left to right), a post-processing could be applied to display the syntactic tree using the sentence direction (e.g., from right to left).

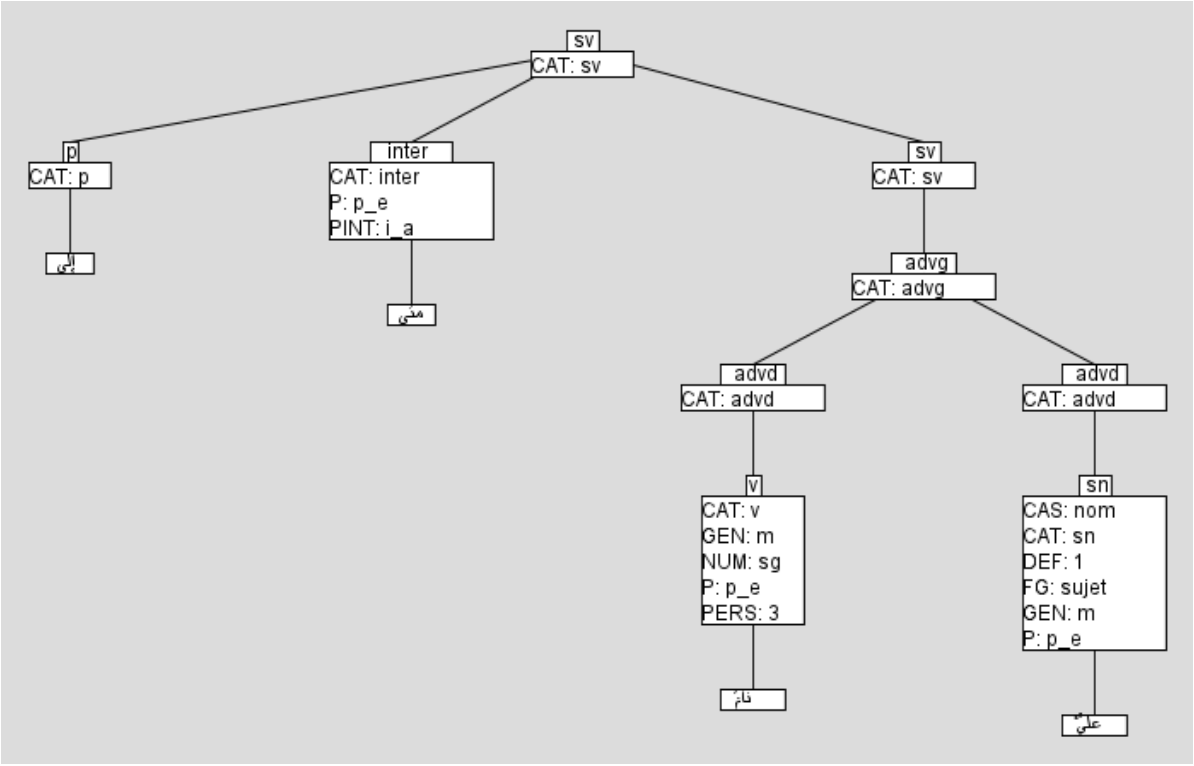


Figure 2: Derived tree for the sentence إلى متى نامَ عليٌّ (*'until when did Ali fall asleep ?'*)

Arabic TreeBank – PATB). The corpus they used is the Part 1 v 2.0 of PATB (Maamouri et al., 2003; Maamouri and Bies, 2004). This extraction involved a reinterpretation of the corpus in dependency structures. The number of elementary trees generated was very high but they did not necessarily offer a good syntactic coverage. In fact, during the process, the authors were able to extract structures with varying positions of the sentence's component (grammatical functions). The resulting combinations are VSO, SVO and OVS. However they could not obtain the VOS combination. This failure is due to the absence of such structures in the corpus used for extraction. Furthermore, the resource is redundant because the researchers manipulated textual forms and not parts-of-speech. As acknowledged by the authors, this extraction was not optimal (grammar cleaning was needed). This somehow advocates for the conjoint use of description languages (that is, not only automatic extraction) to control the output grammar structures.

6 Conclusion and Future Work

In this paper, we showed how to produce a core TAG for Arabic by using the XMG2 system (Petitjean, 2014). First, a modular metagrammatical language for TAG is described by assembling language bricks and the corresponding metagrammar compiler automatically built using the XMG2 system. Then, this metagrammatical language is used to describe TAG trees.

This metagrammatical description benefits from XMG2's high expressivity (e.g. parameterized reusable tree fragments, node identification by means of colors). In particular, we showed how to describe in a relatively concise and yet easily extensible way, simple verbal subcategorization frames in Arabic. Such a description uses a verbal spine containing adjunction points to deal with the various constituent orders in Arabic.

While this work is still in progress, we are considering several extensions of this approach besides improving ArabTAG's coverage. Namely, we plan to integrate morphological and semantic dimensions to ArabTAG borrowing ideas from (Petitjean et al., 2015).

Acknowledgments

We are grateful to Simon Petitjean and three anonymous reviewers for useful comments on this work. This work was partially funded by LIFO.

References

Anne Abeillé, Marie Candito, and Alexandra Kinyon. 1999. FTAG: current status and parsing scheme. In *Proceedings of Vextal-99*, Venice, Italy.

Fériel Ben Fraj. 2010. *Un analyseur syntaxique pour les textes en langue arabe à base d'un apprentissage à partir des patrons d'arbres syntaxiques*. Ph.D. thesis, ENSI La Manouba, Tunisia.

Fériel Ben Fraj. 2011. Construction d'une grammaire d'arbres adjoints pour la langue arabe. In *Actes de la 18e conférence sur le Traitement Automatique des Langues Naturelles*, Montpellier, France, June. Association pour le Traitement Automatique des Langues.

Benoît Crabbé, Denys Duchier, Claire Gardent, Joseph Le Roux, and Yannick Parmentier. 2013. XMG : eXtensible MetaGrammar. *Computational Linguistics*, 39(3):591–629.

Denys Duchier, Brunelle Magnana Ekoukou, Yannick Parmentier, Simon Petitjean, and Emmanuel Schang. 2012. Describing Morphologically-rich Languages using Metagrammars: a Look at Verbs in Ikota. In *Workshop on "Language technology for normalisation of less-resourced languages", 8th SALTMIL Workshop on Minority Languages and the 4th workshop on African Language Technology*, pages 55–60, Istanbul, Turkey.

Dina El Kassas and Sylvain Kahane. 2004. Modélisation de l'ordre des mots en arabe standard. In *Atelier sur le traitement de la langue arabe, JEP-TALN 2004*, page 6. Modélisation de l'ordre des mots en arabe standard. Journées déroulées du 19 au 23 avril à Fès (Maroc).

Roger Evans and Gerald Gazdar. 1996. DATR: A language for lexical knowledge representation. *Computational Linguistics*, 22(2):167–216.

Claire Gardent. 2008. Integrating a unification-based semantics in a large scale Lexicalised Tree Adjoininig Grammar for French. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING'08)*, pages 249–256, Manchester, UK.

Nizar Habash and Owen Rambow. 2004. Extracting a tree adjoining grammar from the penn arabic treebank. *Proceedings of Traitement Automatique du Langage Naturel (TALN-04)*, pages 277–284.

Djamel Kouloughli. 1992. *La grammaire Arabe pour tous*. Press Pocket.

Mohamed Maamouri and Ann Bies. 2004. Developing an arabic treebank: Methods, guidelines, procedures, and tools. In Ali Farghaly and Karine Megerdoomian, editors, *COLING 2004 Computational Approaches to Arabic Script-based Languages*, pages 2–9, Geneva, Switzerland, August 28th. COLING.

Mohamed Maamouri, Ann Bies, Hubert Jin, and Tim Buckwalter. 2003. Arabic treebank: Part 1 v 2.0. LDC Catalog No.: LDC2003T06, ISBN: 1-58563-261-9, ISLRN: 333-321-196-670-5.

Yannick Parmentier, Laura Kallmeyer, Timm Lichte, Wolfgang Maier, and Johannes Dellert. 2008. TuLiPA: A Syntax-Semantics Parsing Environment for Mildly Context-Sensitive Formalisms. In *9th International Workshop on Tree-Adjoining Grammar and Related Formalisms (TAG+9)*, pages 121–128, Tübingen, Germany.

Simon Petitjean, Younes Samih, and Timm Lichte. 2015. Une métagrammaire de l'interface morpho-sémantique dans les verbes en arabe. In *Actes de la 22e conférence sur le Traitement Automatique des Langues Naturelles*, pages 473–479, Caen, France, June. Association pour le Traitement Automatique des Langues.

Simon Petitjean. 2014. *Génération Modulaire de Grammaires Formelles*. Ph.D. thesis, Université d'Orléans, France.

James Rogers and K. Vijay-Shanker. 1994. Obtaining trees from their descriptions: An application to tree-adjoining grammars. *Computational Intelligence, 10:401–421*.

Stuart M. Shieber, Hans Uszkoreit, Fernando Pereira, Jane Robinson, and Mabry Tyson. 1983. The formalism and implementation of PATR-II. In Barbara J. Grosz and Mark Stickel, editors, *Research on Interactive Acquisition and Use of Knowledge*, techreport 4, pages 39–79. SRI International, Menlo Park, CA, November. Final report for SRI Project 1894.

Éric Villemonte De La Clergerie. 2010. Building factorized TAGs with meta-grammars. In *The 10th International Conference on Tree Adjoining Grammars and Related Formalisms - TAG+10*, pages 111–118, New Haven, CO, United States, June.

Fei Xia. 2001. *Automatic Grammar Generation from two Different Perspectives*. Ph.D. thesis, University of Pennsylvania.

XTAG Research Group. 2001. A Lexicalized Tree Adjoining Grammar for English. Technical Report IRCS-01-03, IRCS, University of Pennsylvania.