Développement dâune grammaire computationnelle de lâarabe utilisant le formalisme HPSG

Mahmoud Fawzi MAMMERI
CRSTDLA[1], Alger

Résumé :
Cet article décrit une grammaire computationnelle de l'arabe que nous avons mis au point. Le modèle adopté pour la représentation des signes linguistiques est le formalisme des HPSG (Head-Driven Phrase Structure Grammar), un formalisme linguistique basé sur les contraintes. La grammaire est implémentée sur le système LKB (Linguistic Knowledge Builder), un environnement de développement de grammaire et de lexique pour l'utilisation avec les formalismes linguistiques basés sur les contraintes. Cette grammaire peut servir dans au moins trois domaines. A présent, nous l'utilisons comme grammaire de recherche, surtout en tant qu'outil de validation des hypothèses théoriques. Ce qui fait que les phénomènes à couvrir sont sélectionnés par intérêt linguistique. Elle peut être aussi utilisée comme un excellent outil pédagogique en renfermant les constructions de base de la langue arabe ainsi que les phénomènes les plus usuels. Enfin, elle peut aussi être vue dans une approche MT (Machine Translation). Dans

[1] Centre de Recherche Scientifique et Technique pour le Développement de la Langue Arabe.

cette optique, la grammaire peut faire l'objet d'un point de départ pour le développement d'une grammaire computationnelle bilingue, du fait que les grammaires HPSG/LKB sont utilisées aussi bien dans l'analyse que dans la génération.

1. Motivation

La dernière décennie a vu se développer des implémentations de grammaires à large couverture de plusieurs langues dans différents cadres théoriques. Les cadres théoriques qui ont drainé le plus de travaux sont incontestablement la Head–driven Phrase Structure Grammar (HPSG), la Lexical Functional Grammar (LFG) et la Lexicalized Tree Adjoining Grammar (LTAG). En HPSG, des grammaires assez considérables ont été implémentées pour différentes langues. Cet engouement pour HPSG est du, certes, principalement, au modèle lui-même, mais, aussi, pour les possibilités d'implémentation et d'interfaçage qui l'accompagnent ; surtout avec des systèmes de développement de grammaire très puissants tels que le LKB [Copestake, 2002], et la disponibilité d'une grammaire générique, la LinGO Grammar Matrix [Bender et al., 2002], qui constitue un point de départ unifié pour les différents travaux en ingénierie de grammaire.

2. Introduction

Une grammaire computationnelle est une description grammaticale d'un langage naturel dans un cadre computationnel. Une fois développée, une telle grammaire peut faire partie de divers types d'applications en Traitement Automatique des Langues Naturelles

(TALN) telles que la traduction automatique, les systèmes de question-réponse, etc. Dans cet article, nous allons présenter une première tentative pour le développement d'une grammaire computationnelle de l'arabe. Pour ce faire, nous avons analysé un certain nombre de phénomènes linguistiques de l'arabe dans le cadre du formalisme des Grammaires Syntagmatiques Guidées par les Têtes (Head-driven Phrase Structure Grammar, ou HPSG). Le fragment de grammaire style-HPSG que nous avons obtenu a fait l'objet d'une implémentation dans une plate-forme informatique connue sous le nom de LKB (ou Linguistic Knowledge Builder, pour Base de Connaissances Lexicales).

3. Cadre théorique : Le formalisme des HPSG

La Grammaire Syntagmatique Guidées par les Têtes (ou HPSG, décrite dans Pollard et Sag, 1987, Pollard et Sag, 1994, Sag et Wassow, 1999) est une théorie qui est issue de plusieurs courants théoriques et relève, comme GPSG, des grammaires d'unification. C'est une grammaire bidirectionnelle[2] se proposant de fournir un cadre de modélisation de principes grammaticaux universels. Ce qui différencie HPSG des autres modèles, est sa volonté de donner des descriptions uniformes des différentes dimensions du langage. Cette uniformité de la modélisation se manifeste en ce que le modèle de toute unité est construit sur le même patron quelque soit sa taille. En d'autres termes, utiliser les structures de traits comme cadre unique pour représenter des informations

[2] Les grammaires écrites dans ce modèle sont utilisées en analyse comme en génération.

linguistiques de nature aussi hétérogènes que phonologique, syntaxique, sémantique, etc. Ainsi, un mot (i.e. une unité du lexique) est représenté de la même manière qu'un syntagme ou qu'une phrase, voire un discours ; tous ces objets étant des signes, qui ne sont à leur tour que des structures de traits typées (*TFSs, Typed Feature Structures*). Les règles de grammaires, les principes généraux et les grammaires elles mêmes ne sont que des structures de traits typées.

4. La Plate-Forme d'Implémentation: Le système LKB

Le système LKB (Linguistic Knowledge Builder), conçu par Ann Copestake, est un environnement de développement de grammaire et de lexique open source, implémenté en Common Lisp, et qui a été conçu pour l'implémentation des grammaires à base de contraintes (grammaires de style HPSG). Il utilise une structure de donnée unique, la TFS, pour représenter l'information linguistique, et une seule opération de combinaison, l'unification. C'est un environnement de développement spécialisé de très haut niveau qui peut être utilisé pour le développement de différents types de grammaires ; intégrant des facilités et masquant à l'utilisateur des aspects spécifiques au langage de programmation. [Copestake et al., 2002] En effet, le LKB inclus un analyseur, un générateur, un support pour des hiérarchies d'héritage à grande échelle, divers outils pour la manipulation des représentations sémantiques et un riche ensemble d'outils graphiques pour l'analyse et le débogage de grammaires. En outre, plusieurs grammaires LKB d'assez importantes tailles font partie de divers projets tels que la traduction automatique, l'apprentissage

de grammaires, la traduction automatique du langage parlé, la génération automatique de discours et la réponse automatique d'email.

5. Le langage TDL

Les grammaires LKB sont implémentées en TDL. Le TDL (Type Description Language, ou langage de description de type) est un langage de description qui permet la spécification de TFSs (Typed Feature Structures) ; donc de types, de contraintes, d'entrées lexicales, etc. LKB prévoit l'utilisation d'une variété de langages de description. Le plus communément utilisé est celui adopté par Copestake[3], qui est une version simplifiée de la syntaxe du TDL du système PAGE[4].

6. Implémentation

6.1 Méthodologie de Développement

Pour démarrer la construction d'une nouvelle grammaire LKB, deux choix se présentent actuellement. Le premier consiste en la création d'une nouvelle grammaire en partant de "zéro". Le second est d'utiliser la LinGO Grammar Matrix pour générer automatiquement une 'grammaire de départ'[5] qui consiste au strict minimum nécessaire pour le fonctionnement d'une grammaire dans la langue cible. Pour la grammaire actuelle, nous avons opté pour la première méthode qui nous permettra d'expérimenter plus le système LKB.

---

[3] Copestake A., (2002), *Implementing Typed Feature Structure Grammars*, CSLI Publications, Stanford, Ca.
[4] Uszkoreit et al, 1994.
[5] Starter Grammar.

6.2 La Langue en Question

La version de l'Arabe objet de ce travail est l'Arabe Standard Modern (ASM). L'ASM est la langue de la littérature arabe moderne depuis près d'un demi siècle. Elle est aussi la langue utilisée par les médias. C'est la langue qu'on peut qualifier d'universellement comprise par les locuteurs arabes et c'est aussi la langue enseignée dans les écoles. Les phénomènes linguistiques couvert par notre grammaire, et que nous allons décrire dans ce qui suit, ont été choisi par intérêt linguistique.

6.3 Phénomènes Grammaticaux

6.3.1 Ordre de Mots Elémentaires

L'ordre des mots est relativement libre en arabe. Cette flexibilité est due essentiellement au fait que l'arabe exprime des informations grammaticales à travers la flexion. Cependant, les clauses en *VSO*[6] et *SVO* sont les plus préférées et les plus fréquemment utilisées. Dans l'exemple (1) où l'ordre est VSO, *akala* est le verbe, suivi par le sujet *al-waladu*, ensuite par l'objet *at-tuffaahata*. Le sujet peut précéder le verbe, donnant l'ordre SVO[7]. Dans l'exemple (2), *al-waladu* est le sujet, suivi par le verbe *akala*, et enfin l'objet *at-tuffaahata*.

(1) akala l-walad-u t-tuffaahat-a

[6] VSO, pour Verb Subject Object.

[7] À noter que dans le cas où l'ordre des mots est SVO, il y a une autre analyse possible. Il s'agit de considérer le sujet comme *sujet* ou *thème* (*topic phrase*) et le reste de la phrase comme *propos* (*comment*) dans lequel cas le sujet du verbe est un pronom elliptique qui fait référence au sujet.

eat-PFV.3sg DEF-boy-NOM DEF-apple-ACC
'the boy eaten the apple'

(2) al-walad-u akala al-tuffaahat-a
DEF-boy-NOM eat-PFV.3sg DEF-apple-ACC
'the boy eaten the apple'

Nous avons implémenté la *Head-Complement Rule* décrite dans [Sag & Wassow, 1999] qui a nécessité le développement de deux règles génériques qui combinent les compléments un à un. La première promeut les mots en syntagmes pour qu'ils puissent participer aux combinaisons ; le trait *COMPS* reste inchangé. La seconde permet de licencier le premier complément de la liste *COMPS* et de recopier la liste *COMPS*, sans le complément réalisé, de la tête syntaxique sur le syntagme projeté. Cette dernière règle fonctionne d'une manière récursive comme suit :

```
head-complement-rule-1 := binary-head-initial &
[ SUBJ #subj,
  SPR #spr,
  COMPS #comps,
  ARGS < [ SUBJ #subj & <>,
           SPR #spr,
       COMPS [ FIRST #1, REST #comps ] ], #1 > ].
```

Nous avons aussi implémenté deux règles, en l'occurrence la *Head-Subject Rule* et la *Subject-Head Rule*, pour prendre en charge l'ordre des mots en arabe. Une autre règle la *Head-Modifier-Rule* a été introduite pour rendre compte de la post-modification (par les adverbes, les adjectifs et les syntagmes prépositionnels).

6.3.2 Accord

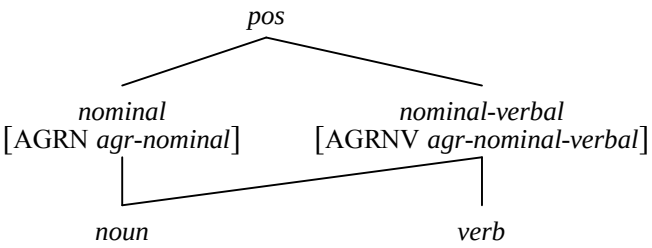
L'accord est défini comme une relation entre des mots partageant un ou plusieurs traits morphosyntaxiques. C'est un trait très puissant en arabe. Comme le souligne Fassi Fehri *" Standard Arabic is an agreement language, with a rich and complex agreement system that interacts with different syntactic elements"*[8]. D'ailleurs, la diversité de la flexion en morphologie arabe est due en grande partie à un besoin d'accord. La langue arabe possède treize traits d'accord distribués en cinq catégories grammaticales : l'*accord en genre* (masc, fem), l'*accord en nombre* (sg, dl, pl), l'*accord en définitude* (definite, indefinite), l'*accord en cas* (nom, acc, gen), l'*accord en personne* (1st person, sd, rd). [Attia, 2008] Les règles d'accord sont rigides et bien définies, et sont de ce fait faciles à coder en TDL. Selon une classification donnée par [Attia, 2008], l'Arabe Standard Moderne possède neuf types d'accord selon le degré d'accord requit. Le type d'accord que nous nous sommes intéressé est l'accord en genre, nombre et personne. Ce degré d'accord est le plus souvent réalisé dans le cas de l'*accord sujet-verbe*, qui a suscité le plus grand nombre de travaux. Pour l'implémentation, nous avons subdivisé le trait traditionnel *AGR* (*AGREEMENT*) en deux sous traits (*AGRN* et *AGRNV*) pour rendre compte du fait qu'il y a des traits qui sont purement nominaux (appropriés au type *agr-nominal*) tandis que d'autres sont appropriés en même temps aux catégories noms et verbes (appropriés

[8] Abdelkader Fassi Fehri, *Issues in the Structure of Arabic Clauses and Words*, Dordrecht: Kluwer Academic Publishers, 1993, p. xi.

au type *agr-nominal-verbal*). Ce qui a donné les déclarations de types suivantes :

```
agr-nominal := feat-struc &
[ CASE case,
  DEF definitude ].
agr-nominal-verbal := feat-struc &
[ NUM num,
  PER per,
  GEND gend ].
```

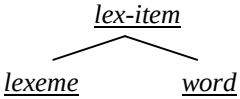
La sous-hiérarchie de type correspondante se présente de manière simplifiée comme suit :



6.3.3 Sous-catégorisation

Pour rendre compte du phénomène de sous-catéorisation, nous allons introduire la notion de *lexème* qui a une grande une importance dans l'organisation de notre lexique. Notre lexique est traité en termes d'une hiérarchie de type. En organisant le lexique de cette manière, nous pouvons exprimer les propriétés partagées des différentes classes de mots. Une fois une hiérarchie lexicale est mise en place, les entrées lexicales que nous écrirons deviennent très simplifiées. Notre conception du lexique suit [Sag & Wassow, 1999]. Nous proposons ainsi le type *lex-item* (*lexical-item*) qui couvre tous les types d'items lexicaux ; c'est le type le plus général de

structures de traits. Ses deux sous-types immédiats sont *lexeme* et *word*. Ces deux types correspondent aux deux différentes utilisations du terme *word* en arabe. En arabe, *kataba*, *ɑ́ktub*, *yaktubuuna*, etc., sont des réalisations différentes du même mot. Elles sont prononcées différemment, elles ont imperceptiblement des sens différents, et ont légèrement des restrictions de cooccurrence différentes. Mais les arabophones n'ont aucune hésitation au niveau compétence à considérer qu'il s'agit de différentes formes du mot *kataba*. Il s'agit manifestement de deux conceptions très différentes d'un même mot : la première se rapporte à un certain couple de son et de sens, alors que la seconde se rapporte à une famille de tels couples. Dans une théorie formelle de grammaire, ces deux concepts ne doivent pas être regroupés. Le type *word* correspond au premier usage (dans lequel *kataba*, *ɑ́ktub* sont des mots distincts). Les entrées lexicales qui donnent lieu aux structures de mots doivent donc être toutes de type *word*. Le type *lexeme* correspond à la deuxième conception du mot, et peut être pensé comme un mot prototype abstrait. Les règles syntaxiques de la grammaire ne manipulent que des objets de type *word*, tandis que la plupart des entrées lexicales de base sont de type *lexeme*. Ces lexèmes servent ainsi comme des atomes à partir desquels toutes les descriptions linguistiques sont construites. Elles doivent donc être transformés en mots avant de pouvoir entrer dans la syntaxe. Cette transformation se fera par le biais d'un mécanisme très important en HPSG que sont les *Règles Lexicales*, qui permettent de dériver des mots à partir des lexèmes. Nous verrons dans la suite du document quelques une de ces règles lexicales.



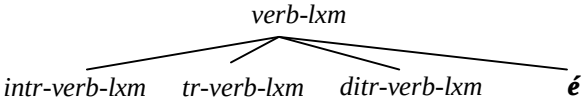
Dans une phrase arabe verbale simple (i.e. construction du genre S → VP), le verbe, qui fonctionne comme la *tête* de la construction[9], sous-catégorise un certain nombre de compléments. Le nombre de compléments dépend de la valence du verbe en question. Un verbe tel que *saaha* (صاح) sélectionne un sujet et ″zéro″ complément. Son entrée lexicale doit, donc, présenter un trait *SUBJ* avec comme valeur une liste de longueur un, à savoir < *NP* >[10], ainsi qu'un trait *COMPS* avec comme valeur une liste vide, à savoir < >. Le verbe *saaha* (صاح) est dit *intransitif* ; il se suffit de son sujet. Le tableau ci-après, donne une description valentielle du verbe en arabe, qui montre que la liste *COMPS* peut comprendre un, deux ou trois compléments.

verbe	sujet	Comp1 ou PP	Comp2	Comp3	PP/ جارومجرور/ظرف/ وصف
saaha	al-waladu				as-saghiiru fii as-sabaahi
yuhibbu	al-waladu	al-mutaala'ata			fii al-layli
hasiba	al-waladu	an-nagaaha	sahlan sahla al-manaali		fii al-'imtihaani al-yawma
a'taa	al-waladu	al-binta	kitaaban		al-baarihata
a'lama	al-waladu	al-binta	al-'ustaada	ghaa'iban ghaa'iban	'ani al-muhaadarati al-yawma
sa'ada	ad-diiku	fawqa as-sathi			sabaahan
saaha	ad-diiku				fawqa as-sathi

[9] Le verbe, à la forme finie, est la tête syntaxique de la proposition déclarative arabe.

[10] La notation <NP> correspond à une liste contenant l'élément NP. < > est la liste vide.

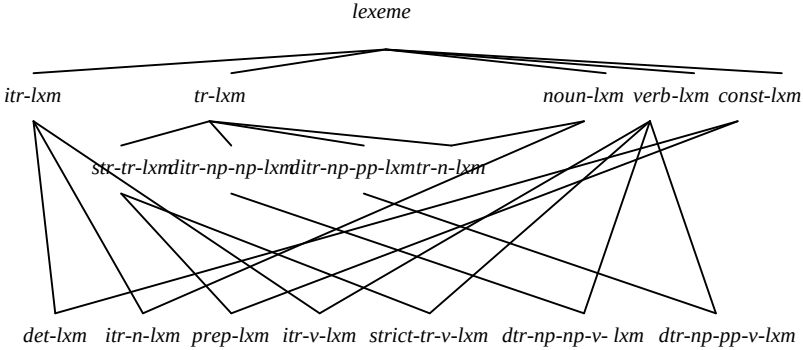
Pour capter un maximum de généralisation au niveau de notre hiérarchie, nous avons essayé de représenter le fait que les noms et les prépositions présentent, eux aussi, des cadres de sous-catégorisation. Pour cela, nous avons implémenté la hiérarchie étendue à héritage multiple en



(4) ci-après à la place de la hiérarchie plate traditionnelle en (3).

(3)

(4)



6.3.4 Marquage de cas

Une langue donnée possède un système de cas si les noms varient dans leurs formes selon le rôle grammatical qu'ils jouent dans la phrase et/ou selon la tête précise dont ils dépendent. L'Arabe présente un système de cas grammatical très intéressant. Le marquage de cas est souvent réalisé par une déclinaison (flexion) spécifique à

la fin du mot. La langue arabe est une *langue nominative-accusative* (*nominative-accusative language*, ou simplement, *accusative language*). Le système de cas en arabe possède trois cas. Le *cas nominatif* qui s'applique plus particulièrement au sujet, et peut être réalisé comme un suffixe ***ï u*** au singulier. Le *cas accusatif* qui s'applique, entre autres, à l'objet, et peut être réalisé comme un suffixe ***ï a*** au singulier. Enfin, le *cas génétif* qui s'applique, entre autres, à l'objet de prépositions, et peut être réalisé comme un suffixe ***ï i*** au singulier.
Au niveau de l'implémentation, le marquage de cas a été codé conformément à un ordre défini pour contraindre les verbes et les prépositions avec les noms qu'ils sous-catégorisent. Les déclarations de types en TDL en (5), (6) et (7) montrent comment les contraintes pour l'accord en cas sont implémentées au niveau du système de typage. (nb. Nous avons allégé les descriptions, au maximum, des spécifications sémantiques)

```
(5) verb-lxm := lexeme &
    [HEAD verb & [AGRNV [GEND #gend]],
     SUBJ <phrase & [HEAD noun & [AGRNV [GEND #gend]]
& [AGRN [CASE nom]], SPR < >, COMPS optional-list]> ].

(6) verb-lxm-ditransitive-np-np := verb-lxm &
    ditransitive-np-np-lxm &
    [COMPS <[HEAD noun & [AGRN [CASE acc]]],
            [HEAD noun & [AGRN [CASE acc]]]>].

(7) prep-lxm := strict-transitive-lxm & const-lxm &
    [ORTH <! #orth !>,
     HEAD prep & [FORM #orth,MOD],
     SPR <>, SUBJ <>, COMPS <[HEAD noun & [AGRN [CASE
gen,DEF def]],]>].
```

6.3.5 Les Règles Lexicales

Une décision majeure que nous aurons à prendre au niveau de la conception de la grammaire dans nos

travaux futurs est de choisir entre un *composant d'analyse morphologique* pour analyser les mots et les identifier comme *racines* et *affixes*, ou bien d'utiliser une base de données pour y mettre les mots en leurs *formes complètes* (ou *fullform*). Chacune de ces techniques ayant ses avantages et ses inconvénients. Pour le moment, nous avons opté pour un lexique génératif. Le fichier LKB de lexique ne représente qu'un sous ensemble du lexique ; la partie la plus importante du lexique LKB est générée automatiquement par le mécanisme des règles lexicales.

Nous avons vu, ci-dessus, qu'au niveau du lexique le type le plus général de structures de traits est *lex-item*, et que ces sous-types immédiats sont *lexeme* et *word*. Les règles syntaxiques de la grammaire ne manipulent que des objets de type *word*, tandis que la plupart des entrées lexicales de base sont de type *lexeme*. Dès lors, ces lexèmes doivent être transformés en mots avant de pouvoir entrer dans la syntaxe. C'est ce qui est réalisé par des règles lexicales (il s'agit, la plupart du temps, d'appliquer une série de règles lexicales) de différents types ; *l2l-lr*, *l2w-lr*, ou *w2w-lr*[11]. C'est ce que nous allons voir ci-après pour les noms propres et les noms communs. D'autres règles ont été prévues pour les autres catégories, i.e. les verbes (des règles lexicales sont nécessaire pour générer les formes féminin, plurielles, …), les adjectifs, etc., et qui non pas été commentées ici pour fautes d'espace.

[11] Abréviations de *lexeme-to-lexeme lexical rule* (en entrée, un objet de type *lexeme* ; en sortie, un objet de type *lexeme*), *lexeme-to-word lr*, *word-to-word lr*.

6.3.5.1 Pour les noms propres

Dans le lexique, les noms propres sont *sous-spécifiés* pour le cas ; ils ne portent pas de désinences casuelles (e.g. zaynab, karīm). Ce type de règles lexicales permet leur flexion. Deux règles morphologiques *nominatif-pn-lr* et *accusatif-pn-lr*, de type *l2l*, ajoutent selon le cas un -u (la voyelle /u/ exprime la marque casuelle du nominatif) ou un -a (la voyelle /a/ exprime l'accusatif), avec instantiation du cas (trait *CASE*). Une troisième règle *nom-acc-pn-lr*, de type *l2m*, s'applique aux lexèmes noms propres, déjà instanciés pour le cas à travers l'une des deux règles précédentes, pour générer un output mot nom propre ; utilisé par exemple dans (a. yā zaydu, b. zaydu 'ibnu 'amī). Il y a lieu de souligner, ici, l'introduction d'un nouveau trait de tête *DEFINDEF* ; son instantiation par cette règle à la valeur *neut* (*neuter*) élimine l'utilisation de l'output dans toutes les autres constructions (autres que a. et b.). La dernière règle *indef-pn-lr*, morphologique, de type *l2m*, permet de générer des noms propres indéfinis, par la suffixation de -n (la marque de l'indéfini, appelée tanwīn, est réalisée par le suffixe /n/, qui est fusionné à celui marquant le cas ; e.g. zayd + u + n ou zayd + a + n).

6.3.5.2 Pour les noms communs

Les noms communs sont sous-spécifiés pour le cas et la définitude ; i.e. ils ne portent ni désinences casuelles, ni ils sont déterminés (e.g. kitāb, walad). Ainsi, des règles analogues ont été implémentées (*nominatif-cn-lr,* ***é***). Plus une nouvelle règle *def-noun-lr* qui permet de générer des noms communs définis. La marque du défini est le préfixe /al/, appelé dans la tradition arabe /al/ de

définition ; il y a donc préfixation de -al et instanciation du trait *DEF* à la valeur *+*.

6.4 Représentation sémantique

Le LKB propose des facilités pour la sémantique. Une fois que la sémantique est intégrée au niveau de la grammaire, le système LKB produit pour chaque phrase analysée une représentation sémantique sous forme de structures MRS. MRS (*Minimal Recursion Semantics*, pour Sémantique à Récursion Minimale) [Copestake et al., 2005] est un cadre pour la sémantique computationnelle. C'est le formalisme standard utilisé à grande échelle dans les grammaires HPSG. MRS n'est pas, en soi, une théorie sémantique à part entière ; c'est un langage de description pour les *formules de la logique du premier ordre* (*First Order Logic*, ou *FOL*).
Dans le but de représenter les *Prédications Elémentaires* (*EPs*) de MRS dans notre interface LKB nous avons créé le type *relation* ci-après qui reflète la structure prédicat-argument.

```
relation := feat-struc &
[ PRED string,
  ARG0 index ].
```

Ainsi que les sous-types *arg1-relation*, *arg1-2-relation* et *arg1-2-3-relation* pour les prédicats d'arités correspondantes :

```
arg1-relation := relation & [ARG1 index].
arg1-2-relation := arg1-relation & [ARG2 index].
arg1-2-3-relation := arg1-2-relation & [ARG3 index].
```

Pour une phrase telle que "*aataa alwaladu albinta altuffaahata qorba almadrasati*", nous obtenons dans les

figures (1) et (2) suivantes (captures d'écrans), respectivement l'analyse de la phrase sous forme arborescente traditionnelle (la TFS correspondante est trop volumineuse pour être reprise ici) ainsi que la structure MRS correspondante.

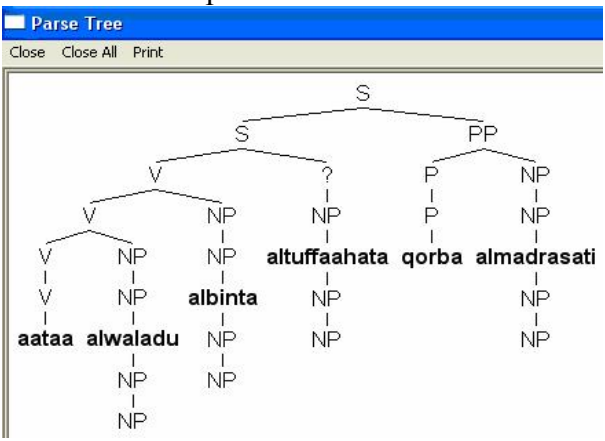


Figure 1 : Analyse arborescente de la phrase "aataa alwaladu albinta altuffaahata qorba almadrasati"

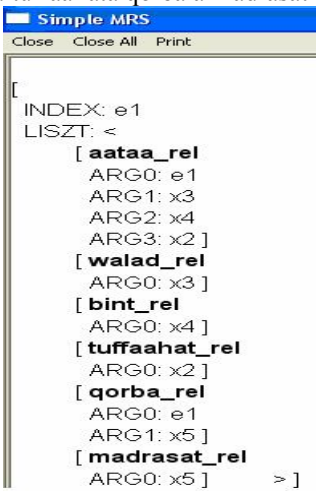


Figure 2 : Structure MRS produite pour la phrase "aataa alwaladu albinta altuffaahata qorba almadrasati"

7. Conclusion et Evolution

La formalisation d'une partie de la syntaxe de la langue arabe nous a permit d'infirmer certaines hypothèses, et de confirmer d'autres. Ce document peut servir comme une première investigation éclairante dans le développement d'une grammaire computationnelle de l'arabe. Nous essayerons dans la prochaine étape de raffiner la modélisation et d'implémenter de nouveaux phénomènes linguistiques. Nous envisageons, dans l'avenir, de continuer le développement de la grammaire dans deux voies parallèles. Nous suivrons le développement de cette grammaire qui servira toujours comme un terrain de test et qui peut faire aussi l'objet d'un cursus pédagogique en Ingénierie de Grammaires. Dans une autre voie, nous essayerons d'utiliser une grammaire connue sous le nom de la Matrix [Bender et al., 2002]. C'est un outil générique qui permet de générer des grammaires adaptées aux langues cibles. Il s'agit d'un utilitaire ('starter-kit') open source pour le développement rapide de grammaires multilingues à large couverture formulée en HPSG et MRS et basée sur le LKB.

Références

Abdelkader Fassi Fehri, *Issues in the Structure of Arabic Clauses and Words*, Dordrecht: Kluwer Academic Publishers, 1993, p. xi.

Attia, Mohammed A., 2008. Handling Arabic Morphological and Syntactic Ambiguity within

the LFG Framework with a View to Machine Translation. PHD thesis, University of Manchester.
Badawi, Elsaid, Carter, M. G., and Gully, Adrian. 2004. Modern Written Arabic, A Comprehensive Grammar. London and New York: Routledge.
Bender E. M., Flickinger D., Oepen S., (2002). *The Grammar Matrix.* In: Proceedings of the Workshop on Grammar Engineering and Evaluation at the 19th International Conference on Computational Linguistics, Taipei, Taiwan, pp. 8-14.
Bender, Emily M. and Dan Flickinger. 2005. Rapid prototyping of scalable grammars. Proceedings of IJCNLP-05 (Posters/Demos), Jeju Island, Korea.
Buckley, R. 2004. Modern Literary Arabic-A Reference Grammar. Beirut: Librairie du Liban.
Carpenter B., (1992). *The Logic of Typed Feature Structures*. Cambridge University Press.
Copestake A., (2002). *Implementing Typed Feature Structure Grammars*, CSLI Publications, Stanford, Ca..
Copestake A., Flickinger D., Oepen S., Malouf R., Carroll J., (2001). *Using an open-source unification-based system for CL/NLP teaching*.

Copestake A., Flickinger D., Pollard C., Sag I. A., (2006). *Minimal Recursion Semantics: An Introduction*, Research on Language and Computation, 3, 281-332.
Flickinger, Dan. 2000. On building a more efficient grammar by exploiting types. Natural Language Engineering 6: 15-28.
Holes, C. 1995. Modern Arabic: Structures Functions and Varieties. London: Longman.
Ibrahim, Khalil. 2002. Al-Murshid fi Qawa'id Al-Nahw wa Al-Sarf [The Guide in Syntax and Morphology Rules]. Amman, Jordan: Al-Ahliyyah for Publishing and Distribution.
Ivan A. Sag and Thomas Wasow., (1999). *Syntactic Theory: A Formal Introduction*. CSLI Publication, Stanford University.
Mammeri M. F., (2003). *Une approche pour l'analyse syntaxique de l'arabe basée sur la Théorie Néo-Khalilienne et HPSG*. Mémoire de Magister, non publié, CRSTDLA-ENSLSH, Alger.
Mammeri M. F., Azzoun H., Zemirli Z., (2006). *Esquisse d'une grammaire HPSG pour l'arabe*. Revue Al-Lisaaniyyaat (revue algérienne de linguistique et des sciences et technologies du langage), N°11, Décembre 2006, Centre de Recherche Scientifique et Technique pour le

Développement de la Langue Arabe, Alger, Algérie.
Mammeri M. F., (2007). *Une grammaire HPSG/LKB pour l'arabe*. A paraître dans la revue Al-Lisaaniyyaat (revue algérienne de linguistique et des sciences et technologies du langage), éditée par le Centre de Recherche Scientifique et Technique pour le Développement de la Langue Arabe, Alger, Algérie.
Pollard C. & Sag I.A., (1994). *Head-Driven Phrase Structure Grammar*, Chicago: University of Chicago Press.
Ryding, Karin C. 2005. A Reference Grammar of Modern Standard Arabic. Cambridge: Cambridge University Press.
Tseng J., (2006). *HPSG for French: Implementation of the Theory*, Loria, Nancy.
Tseng J., (2003). *LKB grammar development: French and beyond*. In F. Fouvry, M. Siegel, D. Flickinger, and E. Bender, editors, Proceedings of the ESSLLI-2003 Work-shop '*Ideas and Strategies for Multilingual Grammar Development*', page 91-97, Vienna.
Sibawaihi, Abu Bishr 'Amr. 1966. Al-Kitab. Cairo, Egypt: Dar al-Qalam.

Wehr, Hans. 1979. A Dictionary of Modern Written Arabic. Ithaca, NY: Spoken Language Services, Inc.
Wright, W. 1896/2005. A Grammar of the Arabic Language. Cambridge: Cambridge University Press.