République Tunisienne
Ministère de l'Enseignement Supérieur

Université de Sfax
Faculté des Sciences Economique et de Gestion

MEMOIRE

en vue de l'obtention du diplôme de

MASTER

en Systèmes d'Information et Nouvelles Technologies

ArabicLDB : une base lexicale normalisée pour la langue arabe

Réalisé par

Aïda KHEMAKHEM

Soutenu le 2 Novembre 2006, devant la commission d'examen

M. Abdelmajid Ben Hamadou	**Président de jury**
M. Gil Francopoulo	**Rapporteur**
M. Bilel Gargouri	**Encadreur**
M. Abdelhamid Abdelwahed	**Co-encadreur**

Année Universitaire
2005-2006

Remerciements

Le travail présenté dans ce mémoire n'aurait pu aboutir sans l'aide et le soutien de nombreuses personnes, je leurs exprime ici mes sincères remerciements et ma profonde reconnaissance. Je tiens à remercier particulièrement :

Mon encadreur, Monsieur Bilel Gargouri pour son encadrement, son soutien sans failles et sa disponibilité. Ses conseils, ses suggestions de lecture, ses commentaires, ses corrections et ses qualités scientifiques ont été très précieux pour mener à bien ce travail.

Mon co-encadreur, Monsieur Abdelhamid Abdelwahed, pour son suivi de mon mémoire, l'attention qu'il y a porté et sa disponibilité. Ses remarques, ses conseils et ses qualités scientifiques m'ont permis d'améliorer la qualité de ce mémoire.

Le responsable de la norme LMF, Monsieur Gil Francopoulo qui s'est intéressé à la langue arabe et d'accepter de participer à mon jury. Je le remercie de ses éclaircissements bénéfiques et de son aide.

Tous mes remerciements également à Monsieur Abdelmajid Ben Hamadou d'avoir accepté d'être président de mon jury et pour l'intérêt qu'il a porté au sujet.

Je n'oublie pas mes enseignants à la faculté des sciences économiques et de gestion de Sfax, mes amis (Fatma, Mehdi, Najla, Hela, Nouha et Mariem) et les membres du laboratoire MIRACL.

Je remercie les membres de ma famille pour leur incessant soutien et plus particulièrement mes parents qui m'ont guidé sur le chemin des études.

Je tiens enfin à remercier chaleureusement mon mari qui m'a aidé et encouragé de terminer la recherche.

A toutes et tous, un grand merci !

Sommaire

Liste des figures

Liste des tableaux

Introduction générale

Le besoin du Traitement Automatique des Langues Naturelles (TALN) en ressources lexicales de grande taille ne cesse de s'intensifier. La gestion de telles connaissances linguistiques doit être prise en considération en priorité puisqu'elle constitue un élément fondamental dans la réussite (dans le sens efficacité) des applications du TALN qui les utilisent. Aussi, engendre-t-elle généralement, une grande partie du coût de ces applications. Ainsi, s'accroît l'intérêt du développement de bases lexicales réutilisables et indépendantes d'une application linguistique particulière.

Les connaissances relatives à une base lexicale sont diverses (i.e., morphologiques, syntaxiques, sémantiques), complexes et de grandes tailles, ce qui a influencé négativement les nombreux projets nationaux et internationaux qui se sont intéressés au développement de bases lexicales (monolingues ou multilingues). En effet, il y a eu une multitude de modèles de bases lexicales, notamment pour les langues latines, avec une absence de norme de gestion et d'échange.

Convaincu de la nécessité d'une normalisation du processus d'élaboration des bases lexicales, certains chercheurs ont pris l'initiative de faire des propositions dans ce sens. Parmi ces propositions, nous citons en particulier le projet Franco-Américain LMF (Lexical Markup Framework) qui a vu le jour en 2003 **[Francopoulo, 2003]**. Ce projet est étudié par le sous comité SC4 du comité technique TC37 de l'ISO, responsable des ressources linguistiques. Le TC37-SC4 a attribué le numéro ISO-24613 au projet LMF **[Francopoulo, 2006a]**, encore dans une version de validation, en attendant de le publier comme norme de gestion et d'échange des ressources lexicales monolingues et multilingues. Se basant sur d'autres standards, notamment l'ISO 12620 relatif aux catégories de données, LMF décrit une démarche qui part d'un modèle abstrait couvrant tous les niveaux des langues sous forme d'un noyau et d'un ensemble d'extensions (i.e., morphologique, syntaxique et sémantique) pour aboutir à une représentation concrète en XML.

Dans sa dernière version, LMF a été testé convenable pour plusieurs langues européennes, asiatiques et américaines. Certaines illustrations ont déjà vu le jour, comme la base lexicale morphologique *Morphalou* pour le Français **[Romary et al, 2004]**. Cependant, les caractéristiques de la langue arabe n'ont pas été confrontées, jusqu'à présent, aux propositions de LMF de manière à valider l'application de cette future norme à la langue arabe, bien que cette langue soit pratiquée par une grande population dans le monde.

Etant conscient de la nécessité de considérer la langue arabe dans les travaux qui se font actuellement au niveau de l'ISO pour la validation de LMF, nous nous proposons, dans le

présent travail, d'étudier les possibilités d'appliquer LMF pour l'arabe. Nos travaux se font dans le cadre d'un projet de coopération Tuniso-Français entre notre laboratoire MIRACL (Multimedia, InfoRmation systems and Advanced Computing Laboratory), l'unité de recherche LSCA (la Linguistique et les Systèmes Cognitifs Associés) du côté Tunisien, et l'équipe Langue et Dialogue du laboratoire Loria/INRIA du côté Français. Signalons que parmi les membres de l'équipe Française, nous trouvons M. Gil Francopoulo, l'un des éditeurs de LMF, et M. Laurent Romary, président du sous-comité TC37-SC4 de l'ISO.

Nous chercherons, dans le cadre du présent travail, à apporter les renseignements nécessaires au sous-comité TC37-SC4 de l'ISO pour prendre en considération les caractéristiques de l'arabe (enrichir le modèle). Par conséquent, nous serons amenés à apporter quelques rectifications (ajouts) au registre relatif à la norme ISO 12620 des catégories de données. Notre objectif sera aussi de développer la base lexicale de l'arabe ArabicLDB qui soit conforme à cette future norme. Notre contribution se limite, à présent, au niveau morphologique de l'arabe, avec la modélisation des paradigmes de flexion verbale associés. Des outils d'acquisition et de recherche, et notamment un conjugueur, supporteront la base *ArabicLDB*.

Ce mémoire s'articule autour de six chapitres et comporte quatre annexes :

Dans le premier chapitre, *état de l'art des bases lexicales*, nous commençons par étudier quelques propositions de bases lexicales existantes (latines et arabes). Ensuite, nous critiquerons ces ressources lexicales et nous montrerons le besoin d'une norme. Nous présenterons à la fin le projet LMF pour la standardisation de la gestion et de l'échange des bases lexicales.

Dans le second chapitre, *les caractéristiques phono-morphologiques de l'arabe*, nous présenterons les caractéristiques phono-morphologiques de la langue arabe. Ensuite, nous décrirons la structure d'un mot arabe, les différentes catégories des mots ainsi que les caractéristiques de chaque catégorie. Puis, nous présenterons les traits morphologiques des verbes et des noms arabes. Enfin, nous donnons une idée sur les règles phonologiques qui interviennent dans la morphologie d'un mot arabe.

Après l'étude de LMF et de la langue arabe, nous nous pencherons au troisième chapitre, *application de LMF à la langue arabe*, sur l'application des propositions de LMF pour cette langue. Nous commencerons par la démarche à suivre qui consiste à sélectionner les catégories de données nécessaires à partir du registre normalisé des catégories de données et ajouter les catégories de données spécifiques à l'arabe. Ensuite, nous traiterons la

structure modulaire proposée, en particulier, l'extension morphologique et l'extension des modes de flexion en précisant les choix linguistiques pour l'élaboration de la base. Enfin, nous clôturerons ce chapitre par critiquer certaines propositions de LMF.

Dans le quatrième chapitre, *les paradigmes de flexion des verbes arabes selon LMF,* nous présentons le principe de conjugaison des verbes arabes qui peuvent avoir des cas particuliers nécessitant l'application des règles phonologiques. Ainsi, nous proposerons un nouveau classement pour les verbes arabes, duquel nous extrairons les paradigmes nécessaires pour plus de 16000 verbes tout en respectant les propositions de LMF. Nous terminerons par critiquer les résultats obtenus et proposer une solution qui représente réellement la langue arabe et valorise les règles phonologiques.

Le cinquième chapitre, *conception de la base ArabicLDB*, sera consacré à la conception de l'interface de la base *ArabicLDB*. Nous préciserons les diagrammes de cas d'utilisation, de classes et de séquences, dans le but de détailler les interactions au niveau de l'interface soit avec l'administrateur de la base, soit avec l'utilisateur.

Le sixième chapitre, *réalisation et jeux d'essai*, fera l'objet d'une description de l'architecture de l'environnement et de la réalisation de l'interface pour alimenter et exploiter la base *ArabicLDB*, tout en précisant les choix technologiques et les modes d'utilisation. Ce chapitre couvre un jeu d'essai de notre système de gestion de la base.

Dans l'annexe A, *liste des nouvelles catégories de données selon le modèle ISO12620*, nous présenterons les nouvelles catégories de données qui sont nécessaires pour la morphologie de la langue arabe qui n'existent pas dans la norme ISO 12 620, tout en respectant la description proposée dans cette norme.

L'annexe B, *squelette de classification des verbes arabes*, comportera le squelette de classification des paradigmes, en présentant les différentes combinaisons possibles des schèmes verbaux et des classes de racine qui servent à l'identification des paradigmes.

L'annexe C, *verbes types des paradigmes*, présentera les verbes types des différents paradigmes identifiés.

L'annexe D, *le DTD de la norme LMF*, sera réservée à la DTD proposée dans la révision 9 de la norme LMF.

CHAPITRE

1

Etat de l’art des bases lexicales

I. Introduction

Les applications du TALN évoluent très rapidement, leurs efficacités dépendent de leur base lexicale, autrement dit, si une base lexicale est riche en informations linguistiques et couvre le maximum d'entrées lexicales, les résultats de l'application seront plus significatifs et plus intéressants. Pour qu'une base lexicale soit robuste et réutilisable, elle doit avoir des traits et des catégories qui couvrent le maximum de niveaux linguistiques (morphologique, syntaxique, sémantique…), avec des nominations qui ont un sens pour un auditoire le plus large possible et une organisation adéquate et simple afin de faciliter son exploitation. En effet, il y a plusieurs projets qui visent l'amélioration des bases lexicales monolingues, bilingues et même multilingues. Parmi ces projets, c'est LMF, le nouveau né de l'ISO qui vise la normalisation des bases lexicales.

Tout d'abord, nous allons présenter les principaux modèles des bases lexicales latines qui sont assez nombreuses et quelques bases lexicales arabes qui souffrent d'un grand manque (insuffisance). Ensuite, nous présenterons la proposition de normalisation LMF qui utilise la norme des catégories de données ISO 12620 pour décorer son méta-modèle noyau et les cinq extensions de TALN, selon une démarche de réalisation.

II. Aperçu sur les bases lexicales non normalisées

La construction de bases lexicales et leur utilisation est en pleine explosion dans le domaine du TALN. Les travaux des années précédentes ont permis de développer plusieurs ressources lexicales ayant des structures différentes et répondant à des besoins différents. Nous allons présenter, en première étape, les modèles des langues latines et, en deuxième étape, deux échantillons de bases lexicales arabes.

1. Les bases lexicales latines

Pour les langues latines, les travaux sont assez nombreux, nous citons à ce propos les principales ressources lexicales qui sont classées par Monsieur Francopoulo selon leur filiation **[Francopoulo, 2004]**.

- **La famille TEI**

Le projet international Text Encoding Initiative (TEI) a commencé en 1987. Parmi ses buts, l'établissement des ensembles de balises utilisées pour marquer la structure et les attributs d'un dictionnaire. Chaque dictionnaire a une structure qui lui est propre et il n'est pas possible de proposer une DTD (Document Type Definition) simple pour coder tous les dictionnaires. Malgré ces difficultés, le chapitre 12 de la TEI-P3 présente un certain

nombre d'éléments qui permettent de coder de manière structurée les entrées d'un dictionnaire.

- **La famille des modèles de Mel'cuk**

Papillon est une base lexicale multilingue à « pivot » qui offre des dictionnaires bilingues ou multicibles. Pour chaque langue, on élabore un dictionnaire strictement monolingue au format DiCo de Polguère et Mel'tchuk **[Polguère, 2000]**, qui est une simplification des structures utilisées dans le DEC (Dictionnaire Explicatif et Combinatoire) et qui sont trop complexes pour être utilisées à grande échelle. Cependant, l'ensemble de dictionnaires monolingues est relié par un dictionnaire pivot de liens interlingues dits aussi « axies ».

- **La famille de Princeton**

WordNet **[Miller et al, 1993]** est une base de données lexicales monolingues développée par des linguistes du laboratoire des sciences cognitives de l'Université Princeton. La composante fondamentale sur laquelle repose le système est le synset (synonym set), qui est un ensemble de synonymes contenant toutes les lexies qui sont des quasi-synonymes ou, du moins, possédant une importante intersection de sens.

Le projet EuroWordNet **[Vossen, 1999]** est un réseau sémantique multilingue de type WordNet avec des liens sémantiques inter-langues.

- **Le modèle EDR**

EDR **[Sérasset, 1994]** est un projet industriel visant la construction d'une base lexicale bilingue spécifiquement destinée au couple japonais-anglais de grande taille. Pour atteindre cet objectif, ils ont simplifié les structures linguistiques présentes dans les dictionnaires, et ils ont exprimé les informations grammaticales selon un ensemble fermé d'attributs.

Les équivalences entre langues n'étant généralement pas parfaites, il existe 5 relations de correspondance: équivalence, sous-relation, super-relation, synonymie, et remarque.

- **Les modèles Européens simples**

Les projets BDLex Celex, Multex et «Multex goes East» sont des bases lexicales monolingues ou multilingue qui traitent principalement le niveau morpho-syntaxique. MULTEXT **[Véronis, 1996]**, à titre d'exemple, est un projet européen visant une base linguistique commune. Ce projet a des objectifs simples : tentative de standardisation des ressources textuelles et des données linguistiques, création des ressources linguistiques informatisées, monolingues et multilingues, ainsi que des outils génériques pour l'annotation et l'exploitation de corpus. La mise au point des étiquettes s'est faite dans une optique de simplicité et de compromis visant à leur réutilisabilité. Mais,

l'inconvénient de cet exemple cité et extrait des modèles européens simples c'est qu'il ne traite pas le niveau sémantique.

- **Les modèles Européens complexes**

Le projet GENELEX **[Zaysser, 1992]** est le modèle original. Mais s'il est le plus puissant parmi les modèles européens, il demeure complexe et non simplifiable. De ce fait, un grand nombre d'acteurs en Europe n'empruntent qu'une petite partie de ces modèles sans en prendre en compte la totalité.

La définition préalable du métalangage descriptif applicable aux dictionnaires monolingues, et la description de l'entrée lexicale dans un contexte sont des propositions intéressantes pour résoudre les problèmes spécifiques que pose la mise en correspondance d'entrées lexicales de deux langues. Cependant, la grande difficulté dans le modèle multilingue de Genelex est l'élaboration et la mise au point du noyau pivot.

2. Les bases lexicales arabes

Pour la langue arabe, les travaux sont limités, nous citons à titre d'exemple DIINAR et le lexique de l'analyseur morphologique MORPH2.

- **DIINAR.1**

DIINAR.1 (DIctionnaire INformatisé de l'ARabe - version1) **[Dichy et al, 2002]** est l'une des bases lexicales les plus importantes dans le domaine de TALN traitant la langue arabe. Elle se caractérise par un bon niveau de couverture, environ 20000 entrées verbales et 39000 entrées nominales.

Les concepteurs de DIINAR.1 organisent des informations de natures diverses en différentes bases de données selon la nature de l'unité lexicale traitée. DIINAR.1 **[Abbes, 2004], [Zaafrani, 2001]** servira principalement à générer un lexique qui sera utilisé par des applications de TALN puisque c'est une représentation intentionnelle qui se base sur des règles. D'autre part, les lexiques générés à partir de DIINAR ont des contenus et des formes différentes. L'environnement de génération permet de répondre à des besoins différents sans toucher à la structure de la base de données lexicale.

- **Lexique de MORPH2**

MORPH2 **[Chaâben et al, 2004]** est un analyseur morphologique pour l'arabe non voyellé élaboré au sein de l'équipe LARIS (LAboratoire de Recherche en Informatique à Sfax – Tunisie). Le lexique utilisé dans MORPH2 est stocké dans quinze fichiers XML de structures différentes.

Ce lexique est un exemple de plusieurs lexiques destinés à des applications particulières de traitement morphologique.

3. Synthèse

Des années d'expérience dans le TALN ont engendré des ressources lexicales de structures différentes. Toutes ces propositions cherchent la bonne structure et obligent les utilisateurs à suivre un modèle et un format de représentation, mais malheureusement elles ne sont pas flexibles. L'enrichissement de cette base par des informations d'ordre linguistique engendre la modification de sa structure.

La réutilisation des ressources lexicales est difficile, à cause de la variation de leurs structures et de leurs descripteurs linguistiques. Encore l'échange de données entre les ressources de familles différentes est très difficile.

Ainsi, afin de faciliter les échanges dans la communauté, il est important de mettre en place une norme au niveau conceptuel qui permet la fusion, la réutilisation et l'interopérabilité de ces ressources.

III. La norme LMF : Lexical Markup Framework-ISO 24 613

1. Présentation générale

LMF est une norme de spécification des lexiques monolingue et multilingue destinées au TALN **[Francopoulo et al, 2006a]**. Elle est en cours de validation par le sous comité TC 37/SC 4 responsable des ressources lexicales, sous le numéro 24 613. Un nouveau projet est entamé suite à une proposition américaine au cours de l'été 2003. Il vise l'élaboration de cette norme, couplée à une deuxième proposition pour représenter les catégories de données lexicales (comme partie additionnelle de l'ISO 12620). Néanmoins, la faiblesse de la proposition américaine est consolidée par une proposition française, celle de Gil Francopoulo. Actuellement, c'est la version 9 qui est adoptée. C'est selon cette nouvelle version que nous avons conçu notre travail.

LMF se présente sous forme d'un méta-modèle noyau et d'un ensemble de cinq extensions. La partie noyau spécifie les notions de lexique, de mot, de forme et de sens. Les extensions traitent la morphologie, les paradigmes de flexion, la syntaxe, la sémantique et des notations multilingues. Les principales caractéristiques de cette norme sont la généricité et l'extensibilité. La généricité nous permet de générer des modèles décorés par des catégories de données, à partir du méta-modèle proposé. Ce qui va engendrer l'apparition de plusieurs lexiques conformes à LMF, ayant des catégories de données différentes. En plus, l'extensibilité nous permet de construire des lexiques par la sélection des sous-ensembles de ces extensions possibles qui sont pertinents à nos

besoins. Ces lexiques peuvent être étendus à tout moment par les autres parties des extensions.

2. Utilisation des catégories de données normalisées-ISO 12620

Les catégories de données sont des descripteurs linguistiques élémentaires qui permettent de décorer les classes du méta modèle (i.e., ***/root/***, ***/grammaticalGender/***). Leur gestion est indépendante de leur association effective avec ce méta modèle. Leur normalisation suit des principes définis par la norme ISO 12620. Elles sont organisées dans un registre de catégories de données (RCD) **[Romary et al, 2003]** accessible en ligne (http://syntax.inist.fr/).

D'après la description de ISO 1179, il existe deux genres de catégorie de données (CD) : complexes comme /grammatical gender/ et simple comme /masculine/. Une CD complexe utilise un ensemble de CD simples comme des valeurs possibles.

Tous les attributs de LMF sont des catégories de données complexes. Chaque valeur d'un attribut correspond à une catégorie de données simple ou à une chaîne de caractère (string) Unicode.

La Figure 1 ci-dessous, donne un exemple de représentation d'une catégorie de données complexe selon cette norme **[Romary, 2003]**.

Entry Identifier : gender
Profile: morpho-syntax
Definition (fr): Catégorie grammaticale reposant, selon les langues et les systèmes, sur la distinction naturelle entre les sexes ou sur des critères formels (Source: TLFi)
Definition (en): Grammatical category…
Conceptual Domain: {/feminine/, /masculine/, /neuter/}

Object Language: fr
Name: genre
Conceptual Domain: {/feminine/, /masculine/}

Object Language: en
Name: gender

Object Language: de
Name: Genus
Conceptual Domain: {/feminine/, /masculine/, /neuter/}

Figure 1 : Exemple d'une catégorie de données

Un profil est un ensemble de catégorie de données dans le RDC. Actuellement, il y a un profil pour la terminologie défini par le TC37/SC3 et trois autres profils pour TALN définis par le TC37/SC4 : méta-données, morpho-syntaxique et sémantique. Dans le profil morpho-syntaxique, on s'est limité, pour le moment, à quelques traits représentés dans la Figure 2 qui suit **[Francopoulo et al, 2006b]**:

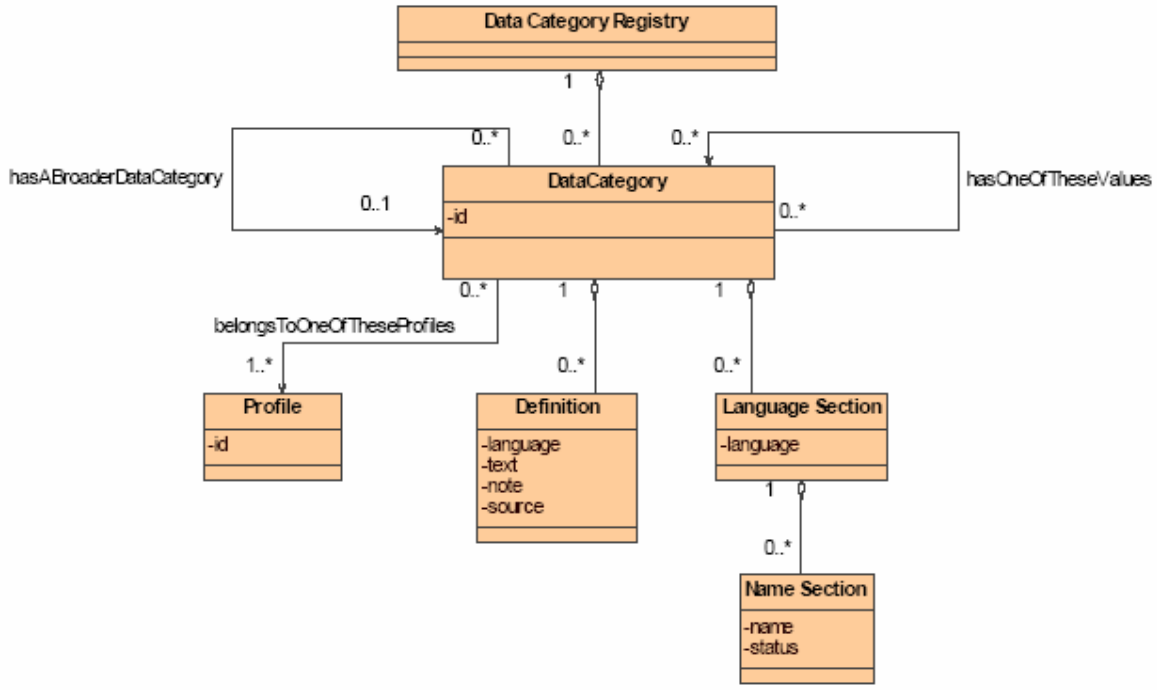


Figure 2 : Les traits du profil morpho-syntaxique

Tout d'abord, il faut différencier entre la notion de relation /broader/ et la notion /conceptual domain/. Le /broader / permet un lien hiérarchique entre deux constantes pré-définies. Par exemple dans la Figure 3, le nom commun est une valeur plus spécialisée que le nom.

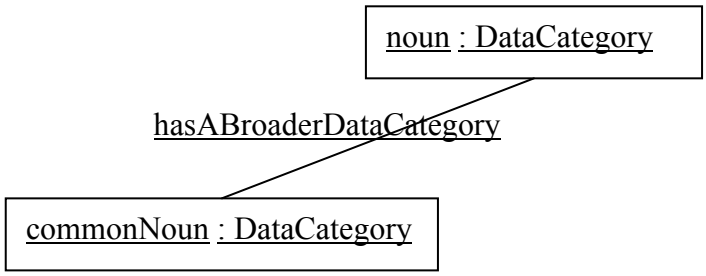


Figure 3 : Exemple de spécification

Mais, la notion de /conceptual domain/ permet à un ensemble de valeurs valides d'être identifié. Par exemple dans la Figure 4, le nom est une valeur pour partOfSpeech.

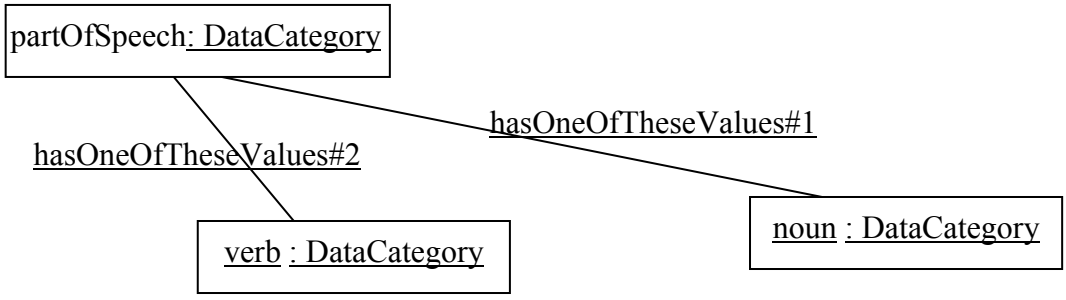


Figure 4 : Exemple d'identification

Enfin, après la sélection des catégories de données et éventuellement l'ajout d'autres catégories, le développeur doit publier les siennes, déjà sélectionnées, et discuter ses nouvelles propositions pour les ajouter au RDC.

3. Le noyau de LMF

Le méta-modèle noyau de LMF est organisé comme une structure hiérarchique des classes UML présentées dans la Figure 5.

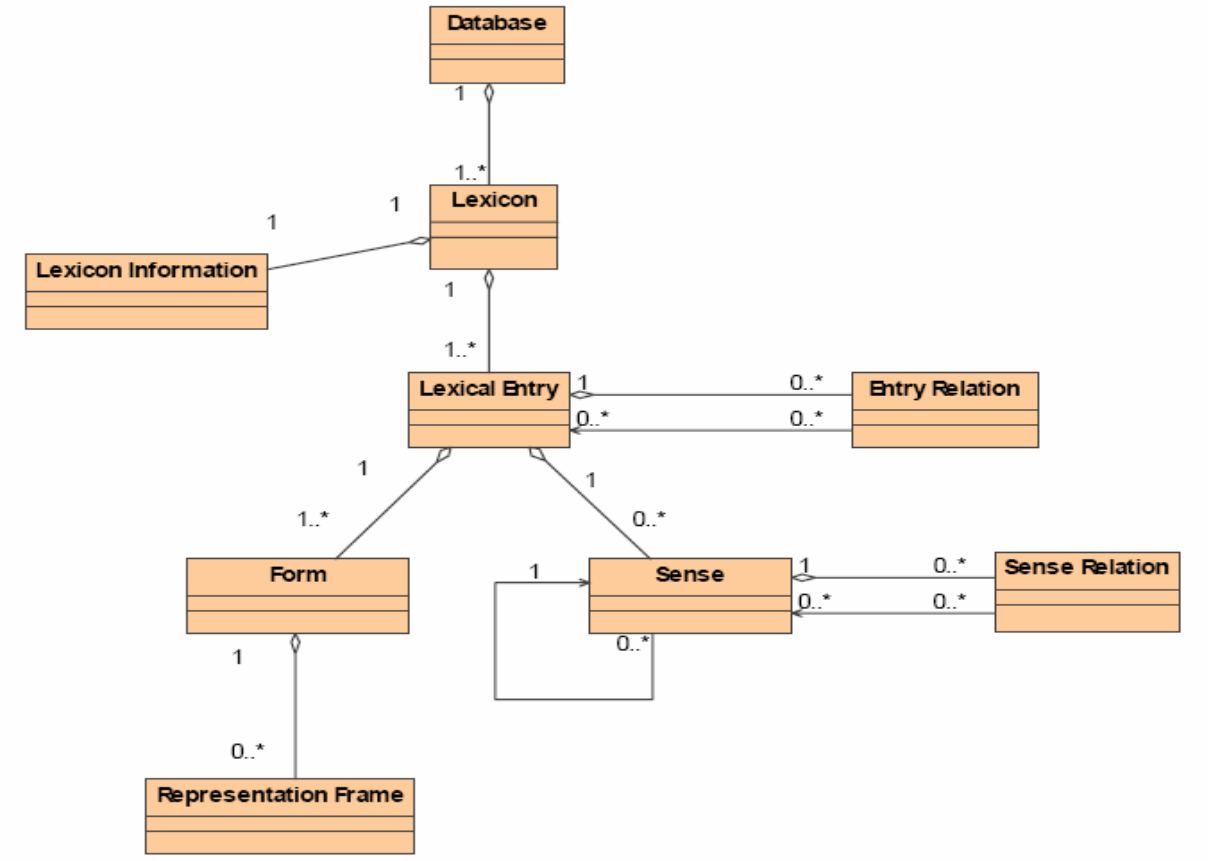


Figure 5 : Le modèle noyau de LMF

Ce méta-modèle comporte les neuf classes suivantes :

- *Database* : la totalité de la ressource qui peut avoir un ou plusieurs *lexicon*.
- *Lexicon Information* : les informations administratives et autres attributs généraux concernant cette base de données tel que le numéro de la version, l'auteur etc.
- *Lexicon* : un lexique d'une langue donnée qui appartient à une seule *Database* et qui peut avoir un ou plusieurs *Lexical Entry.*
- *Lexical Entry* : représente un mot, un mot-multiple ou un affixe de la langue courante. C'est l'unité élémentaire dans une base lexicale qui porte l'information d'une partie du discours. Elle peut avoir un ou plusieurs *Form* et zéro ou plusieurs *Sense*.

▪ *Entry Relation* : présente une relation entre plusieurs entrées lexicales appartenant au même *lexicon* et qui peut avoir des attributs pour décrire cette relation.

▪*Sense* : les attributs qui décrivent le sens du mot. Cette classe peut être partagée par plusieurs entrées lexicales. Cette classe a une relation réflexive et qui est composé par zéro ou plusieurs relations sémantiques.

▪ *Sense Relation* : les attributs qui décrivent une relation entre deux sens à l'intérieur d'une langue. Elle peut être liée au *Sense* à travers la composition ou la référence.

▪*Form* : les valeurs orthographiques et phonologiques des unités lexicales avec des spécifications grammaticales. Cette classe peut avoir deux sous-classes de spécification : *lemmatisedForm* et *inflectedForm* présentées dans la Figure 6.

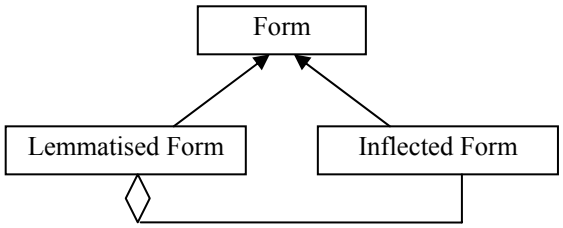


Figure 6 : Les sous classes de Form

- *LemmatisedForm :* une classe de spécification qui hérite toutes les propriétés de la classe *Form* et représente le lemme de cette entrée lexicale. Elle participe avec une seule *Lexical Entry*. Elle peut avoir zéro ou un paradigme et zéro ou plusieurs *InflectedForm*.
- *InflectedForm* : une forme fléchie correspond à une forme d'occurrence d'une *lemmatisedForm*.

▪*Representation frame* : spécifie la représentation orthographique d'un mot s'il en a plusieurs telle qu'une graphie ou une transcription phonologique.

Dans la Figure 1, une ressource *Database* peut avoir un ou plusieurs *Lexicon* qui est composé de plusieurs *LexicalEntry*. Cette entrée lexicale peut avoir une ou plusieurs *Form* et zéro ou plusieurs *Sense*. Ces deux classes nous permettent de greffer les extensions, par exemple l'extension morphologique se greffe sur la classe *Form.*

4. Les extensions

Les fondateurs de LMF ont fait reposer le TALN sur cinq extensions. Ils ont utilisé les classes UML au niveau de la conception. Les classes dont la couleur est blanche appartiennent au noyau, celles dont la couleur est grise font partie de l'extension en cours.

4.1. Extension morphologique

L'extension morphologique est la partie obligatoire pour la plupart des applications de TALN. Cette extension est traitée de deux manières différentes dans LMF. La première présente les formes fléchies, la deuxième fait référence aux paradigmes de flexion pour les générer. Les différentes classes relatives à cette extension sont présentées dans la Figure 7 ci-dessous.

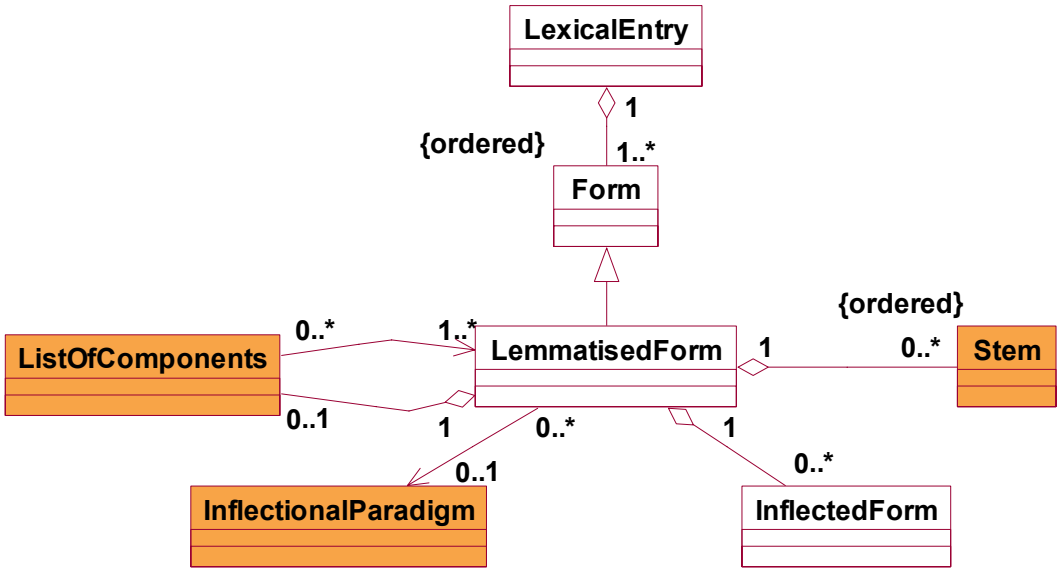


Figure 7 : L'extension morphologique

Les trois classes de cette extension sont :

- *InflactionalParadigm* : factorisation d'un ensemble de structures communes à un grand nombre de mots. Par conséquent, un paradigme peut être utilisé avec plusieurs *LemmatisedForm* .
- *Stem* : représente un des éléments qui forme un mot (par exemple anticonstitutionnellement a deux Stems qui sont respectivement anti- et constitution). Elle peut participer avec un seul lemme qui peut avoir zéro ou plusieurs *Stem* qui.
- *List of Components* : indique l'ordre des mots dans le cas d'un mot multiple.

4.2. Extension des modes de flexion : les paradigmes de flexion

Cette extension permet la description des paradigmes qui permettent la génération des formes fléchies. Un paradigme de flexion comporte des *MorphologicalFeatures Combinators. Un MorphologicalFeaturesCombinator* est une classe abstraite qui fait le lien entre des traits morphologiques et un ou plusieurs *InflectedFormCalculators* qui regroupent des *Operations* et des *OperationArguments* permettant de calculer les formes fléchies correspondantes.

Les opérations sont utilisées une seule fois dans le calcul d'une forme fléchie et les types de ces opérations sont limités : ce sont : /*add before*/, /*remove after*/, /*add after*/, /*substitute*/, /add/, /remove/ et /*copy*/.

Ces différentes classes sont présentées dans la Figure 8 suivante :

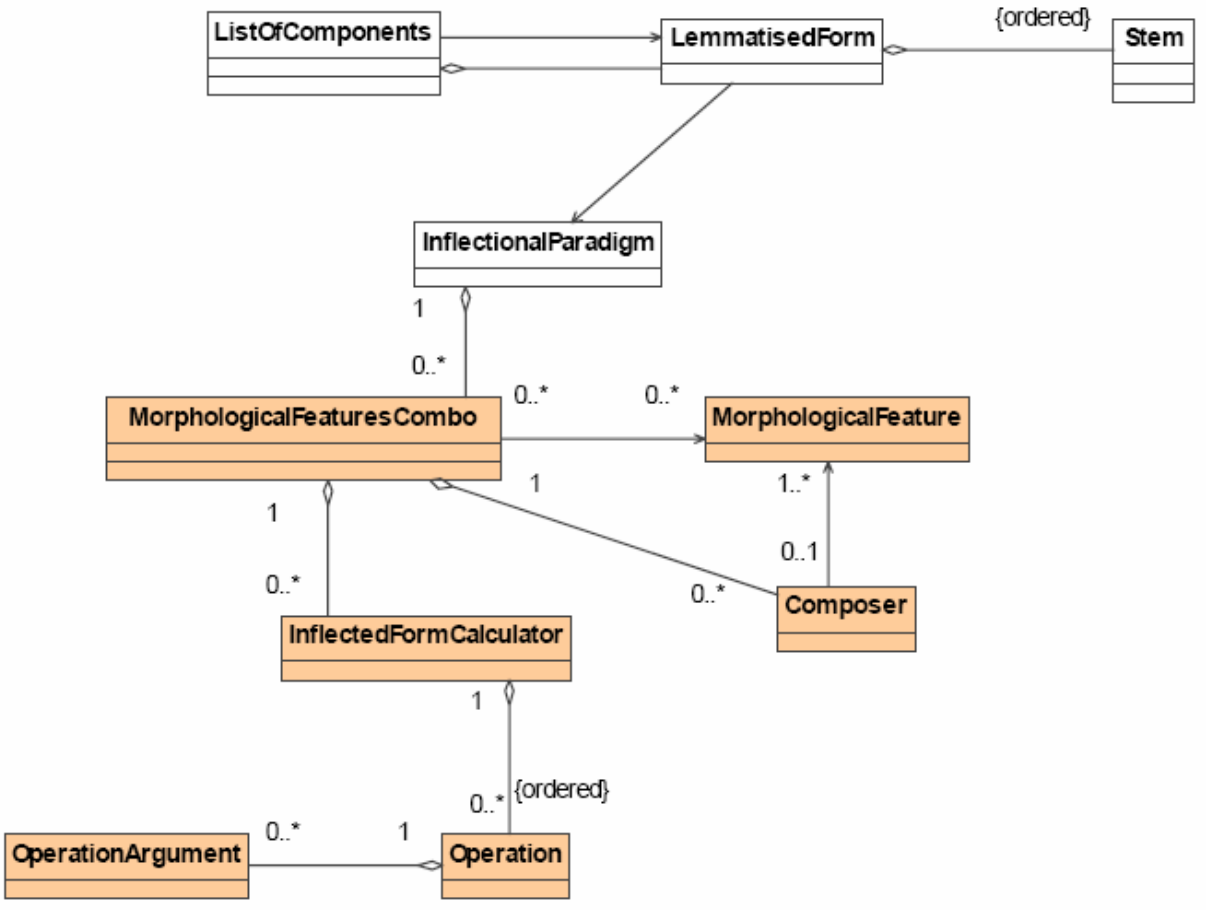


Figure 8 : Le modèle des paradigmes de flexion

Les six classes de cette extension sont :

- *MorphologicalFeaturesCombinator* : permet la combinaison entre des traits morphologiques et les calculateurs d'une forme fléchie. Par conséquent, elle fait référence à zéro ou plusieurs *MorphologicalFeature* et elle se compose par zéro ou plusieurs *InflectedFormCalculator*.
- *MorphologicalFeature* : il s'agit d'un trait qui offre une information linguistique pour une forme fléchie ou un mot. Il est classé parmi les catégories de données traitées par la norme ISO 12620.

▪ *InflectedFormCalculator* : regroupe un ensemble d'opération avec un ordre précis. Chaque opération n'est utilisée qu'une seule fois. Elle appartient à une seule *MorphologicalFeaturesCombinator.*

▪ *Operation* : peut être phonologique ou graphique. Elle associe une liste ordonnée d'arguments.

▪ *OperationArgument* : est associée à une opération. Elle peut être de type textuel ou positionnel. Elle peut être associée à une seule opération.

▪ *Composer* : permet la liaison entre des mots pour former un mot multiple.

4.3. Extension syntaxique

Cette extension est optionnelle, elle est liée à deux classes du noyau à savoir *LexicalEntry* et *Sense,* et à la classe de l'extension sémantique *SemanticArgument.* L'interaction entre ces classes avec les classes de cette extension est présentée dans la Figure 9.

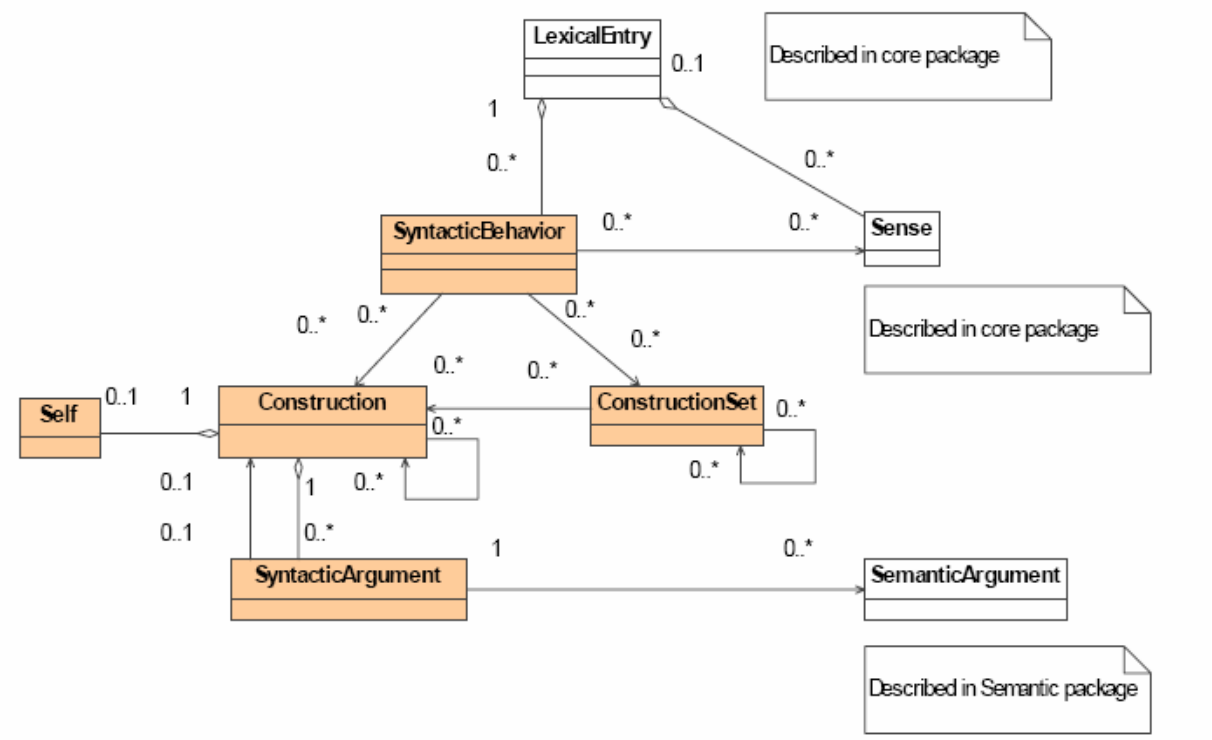


Figure 9 : Le modèle syntaxique

Les cinq classes de cette extension sont :

▪*SyntacticBehavior* : représente un des comportements possibles d'un ou de plusieurs sens.

▪*Construction* : est partagée par tous les mots ayant le même comportement syntaxique dans la même langue. Il peut hériter des relations et des attributs d'une autre

Construction plus générique par la relation de réflexion. Ainsi, il est possible d'intégrer une ontologie hiérarchique des constructions.

- *Self* : se réfère à l'entrée lexicale courante.
- *ConstructionSet* : regroupe un ensemble de construction syntaxique et une relation possible qui subit ces constructions. Il peut hériter des relations et des attributs d'une autre *ConstructionSet* plus générique par la relation de réflexion. Là, également, il y a lieu d'intégrer une ontologie hiérarchique des ensembles de constructions.
- *SyntacticArgument* : décrit un actant syntaxique et peut être lié récursivement à une *Construction* pour décrire des arguments très complexes. Il permet la connexion avec un actant sémantique par *SemanticArgument.*

4.4. Extension sémantique

Cette extension assure les liens entre des définitions, des exemples, des synsets et des représentations prédicatives qui vont permettre la liaison entre les arguments sémantiques et les arguments syntaxiques et qui sont présentés dans la Figure 10.

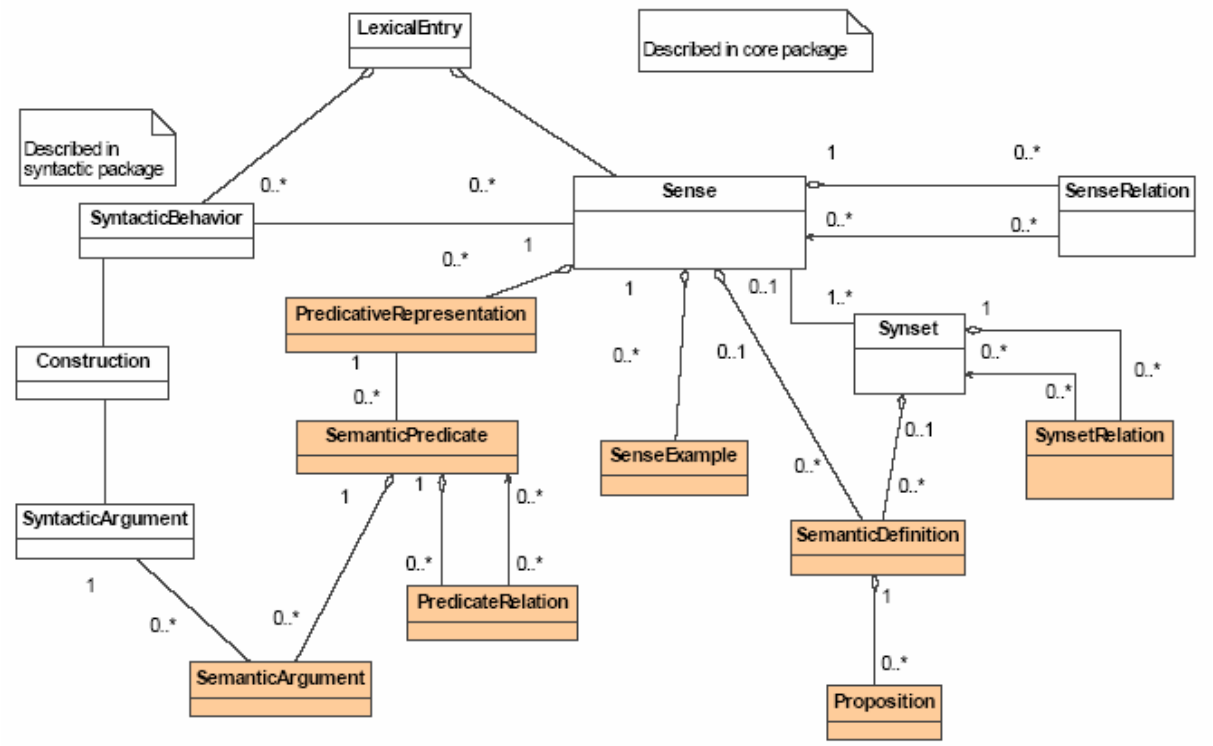


Figure 10 : Le modèle sémantique

Les neuf classes de cette extension sont :

- *SenseExample* : décrit les utilisations d'une signification particulière.

- *SemanticDefinition* : permet la description narrative d'un *Sense* ou d'un *Synset*. Il n'est pas prévu pour l'usage d'un programme, mais il est fourni pour faciliter l'entretien humain.
- *Proposition* : assure le raffinement de la définition sémantique.
- *SemanticPredicate* : peut être utilisé pour représenter une signification commune entre des sens différents qui ne sont pas complètement des synonymes. Ces sens peuvent être liés à des entrées lexicales qui ont des parties de discours différentes.
- *PredicativeRepresentation* : décrit le lien entre le sens et le prédicat sémantique.
- *SemanticArgument* : est une classe consacrée pour le lien d'un actant sémantique avec un actant syntaxique exprimé par le moyen d'un *SyntacticArgument.*
- *PredicateRelation* : permet la description d'une relation entre deux ou plusieurs prédicats sémantiques.
- *Synset* : décrit une signification commune et partagée dans une même langue, autrement dit, il lie des synonymes.
- *SynsetRelation* : permet le lien entre deux *Synsets.*

4.5. Extension des annotations multilingues

Cette extension se base sur des liaisons inter-langues qui vont relier des sens par un sens intermédiaire *SenseAxis.* Le mécanisme de transfert permet de relier deux comportements syntaxiques de langues différentes par la classe *TransferAxis* présentée dans la Figure 11 :

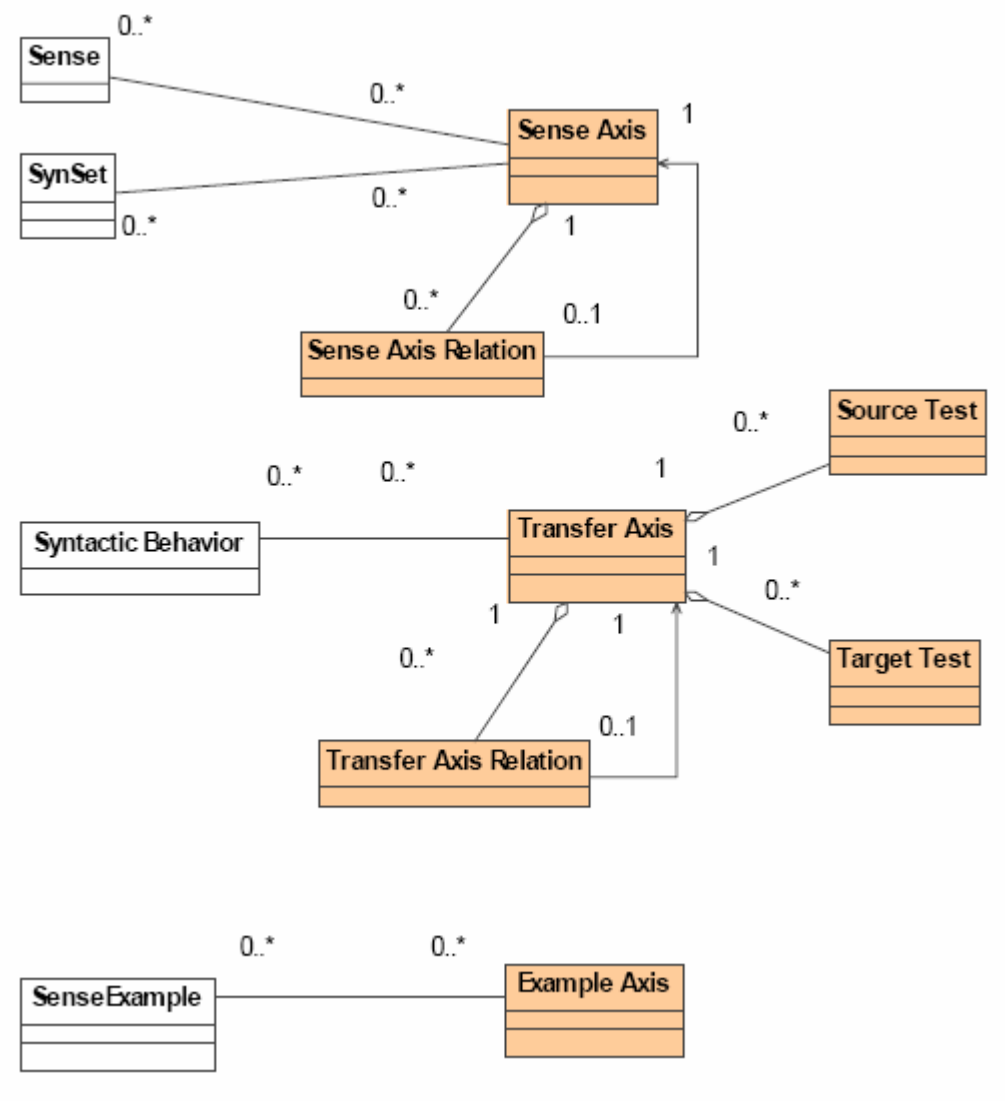


Figure 11 : Le modèle des annotations multilingues

Les sept classes de cette extension sont :

- *SenseAxis* : permet la liaison entre des sens de plusieurs langues différentes. Elle joue le rôle d'intermédiaire.
- *SenseAxisRelation* : décrit la relation entre deux *SenseAxis.*
- *TransferAxis* : est une classe intermédiaire « pivot » qui relie deux comportements syntaxiques de langues différentes.
- *TransferAxisRelation* : relie deux *TransferAxis*.
- *SourceTest* : exprime la condition qui permet la traduction vers la langue source.
- *TargetTest* : décrit la condition qui permet la traduction vers la langue cible.
- *Exemple Axis* : représente des exemples.

IV. Conclusion

Les fondateurs de LMF ont profité de l'expérience des projets précédents (Genelex, Eagles…) pour les langues indo-européennes. Ils ont proposé un modèle qu'ils considèrent complet, générique et extensible se basant sur d'autres normes tels que ISO 12620. Il est à noter que LMF a été testée conforme pour plusieurs langues. En effet, à l'origine Lexical Markup Framework est une proposition franco-américaine. Elle correspond par conséquent au caractéristique du français et de l'anglais. Par la suite, la conformité d'autres langues a été aussi testée (l'italien, l'espagnol…). C'est ainsi que d'autres bases conformes à LMF ont vu le jour à l'instar de *Morphalou* **[Salmon-Alt, 2004]**. En effet, il faut en étudier les possibilités d'application avec d'autres langues qui possèdent des structures différentes de celle de l'indo-européenne tel que l'arabe

CHAPITRE

2

Les caractéristiques phono-morphologiques de l'arabe

I. Introduction

La langue arabe est une langue dérivationnelle et flexionnelle. A l'origine, la langue arabe est la langue parlée par les Arabes. En plus, elle est la langue sacrée du Coran et de l'Islam. Du fait par la propagation de l'Islam et la diffusion du Coran, cette langue est devenue une langue liturgique. Elle est parlée dans 22 pays alors que le nombre de ses locuteurs est plus de 280 millions.

Dans ce chapitre, nous commencerons par présenter les caractéristiques morphologiques de la langue arabe. Ensuite, nous décrirons le mécanisme de dérivation d'un mot arabe. Puis, nous présenterons les différentes catégories grammaticales. Aussi, nous préciserons les traits morphologiques des verbes et des noms arabes. Enfin, nous donnerons un aperçu sur l'influence de la phonologie sur la morphologie de la langue arabe.

II. Les caractéristiques phono-morphologiques d'un mot

La langue arabe s'écrit et se lit de droite à gauche. Le mot arabe s'écrit avec des consonnes et des voyelles. Les consonnes changent de forme de présentation selon leur position dans le mot (au début, au milieu ou à la fin). Les voyelles sont de deux types : les voyelles brèves et les voyelles longues. Elles sont nécessaires à la lecture et à la compréhension correcte d'un texte et permettent de différencier des mots ayant les mêmes consonnes. Malgré l'importance de ces voyelles brèves, elles sont absentes dans la majorité des textes arabes ce qui peut engendrer des ambiguïtés de prononciation et de compréhension.

1. Les consonnes

L'alphabet de la langue arabe comprend vingt-huit consonnes (voir Tableau 1) fondamentales, mais il y a des auteurs qui traitent la lettre *alif* ا comme la vingt-neuvième consonne. L'*alif* se comporte comme une voyelle longue qu'on ne trouve jamais en tant que consonne de la racine.

Parmi ces 28 consonnes, il y a deux symboles (ي، و) qui sont des semi-consonnes (glides), autrement dit, ils peuvent être considérés comme des consonnes ou des voyelles longues (voir paragraphe suivant) selon leur contexte d'apparition, par exemple, ي [y] est une consonne dans نَسِيَ [nasiya] et voyelle longue dans جَمِيلٌ [jamiilun]. Ces semi-consonnes sont classées parmi les consonnes défectueuses حروف العِلّة, leur présence dans un verbe rend ce verbe défectueux (voir paragraphe IV.1).

La représentation graphique des consonnes est différente selon leur position dans le mot, ce qui engendre l'apparition de 100 graphies à partir des 28 consonnes (voir Tableau 1).

Forme	Graphie selon la position			Transcription
Isolée	Initiale	Médiane	Finale	
ء	ء ,أ ,إ ,ؤ ,ئ ,ئ			’
ب	بـ	ـبـ	ـب	b
ت	تـ	ـتـ	ـة ,ـت	t
ث	ثـ	ـثـ	ـث	ṯ
ج	جـ	ـجـ	ج	ǧ
ح	حـ	ـحـ	ح	ḥ
خ	خـ	ـخـ	خ	ḫ
د	د	ـد ,د		d
ذ	ذ	ـذ ,ذ		ḏ
ر	ر	ـر ,ر		r
ز	ز	ـز ,ز		z
س	سـ	ـسـ	ـس	s
ش	شـ	ـشـ	ـش	š
ص	صـ	ـصـ	ـص	ṣ
ض	ضـ	ـضـ	ـض	ḍ
ط	طـ	ـطـ		ṭ
ظ	ظـ	ـظـ		ẓ
ع	عـ	ـعـ	ـع	‘
غ	غـ	ـغـ	ـغ	ġ
ف	فـ	ـفـ	ـف	f
ق	قـ	ـقـ	ـق	q
ك	كـ	ـكـ	ـك	k
ل	لـ	ـلـ	ـل	l
م	مـ	ـمـ	ـم	m
ن	نـ	ـنـ	ـن	n
ه	هـ	ـهـ	ـه	h
و	و	ـو ,و	ـو	w, û
ي	يـ	ـيـ	ـي	y, î

Tableau 1 : Les consonnes de la langue arabe

Toutes les consonnes se lient entre elles sauf (و ، ر ، ز ، د ، ذ) celles qui ne se joignent jamais à gauche. En plus, on peut trouver d'autres représentations qui sont le résultat de concaténation de deux consonnes par exemple, lorsque une ل *lâm* est suivie d'une أ *hamza*, les deux lettres sont remplacées par la ligature لأ.

Šadda est un signe qui peut être placé au-dessus d'une consonne mais qui ne peut pas être à la position initiale du mot. La consonne surmontée de ce signe est analysée comme une séquence de deux consonnes identiques géminées.

2. Les voyelles

Les voyelles ne sont pas comme les consonnes, elles sont rarement notées. Elles sont écrites seulement pour lever des ambiguïtés, dans les éditions du Coran ou dans les ouvrages didactiques. En effet, les voyelles jouent un rôle important dans les mots arabes, non seulement parce qu'elles enlèvent l'ambiguïté, mais aussi parce qu'elles donnent la fonction grammaticale d'un mot indépendamment de sa position dans la phrase. Autrement dit, les voyelles ont une double fonction : l'une est morphologique ou sémantique et l'autre est syntaxique. La langue arabe a deux séries de voyelles, les unes brèves et les autres longues.

2.1. Les voyelles brèves

Les voyelles brèves (ـَ, ـِ, ـُ) sont ajoutées au-dessus ou au-dessous des consonnes. Lorsque la consonne n'a aucune voyelle, on marquera une absence de voyelle représentée en arabe par une voyelle muette (ـْ) comme le montre le Tableau 2.

Voyelle brève	Nom	Transcription
ـَ	فَتْحة /fathatun/	**a**
ـِ	كَسْرَة /kasratun/	**i**
ـُ	ضَمَّة /ḍammatun /	**u**
ـْ	سُكُون /sukûnun/	**-**

Tableau 2 : Les voyelles brèves

Notons que, le concept de "***Tanwin"*** considéré par quelques auteurs comme étant le double de deux voyelles brèves, peut être sous trois formes (ـً, ـٍ, ـٌ) qui sont construit par dédoublement des voyelles brèves. Il est ajouté seulement à la fin des mots indéterminés, par conséquent il n'apparaît jamais avec l'article de détermination اَلْ. Le signe du *tanwin* « ـً » (à l'accusatif) est suivi toujours par ا *alif.*

2.2. Les voyelles longues

Les voyelles longues sont des lettres prolongées, elles sont formées par une des voyelles brèves et une des lettres suivantes (ا, و, ي) comme le montre le Tableau 3 :

Voyelle longue	Transcription
ـَا	â
ـِي	î
ـُو	û

Tableau 3 : Les voyelles Longues

III. Mécanisme de dérivation

En arabe, la majorité des mots (appelés aussi lemme) sont construits sur la base d'une racine tout en respectant un schème : ceci concernant notamment les verbes, les noms et quelques particules.

1. La racine : الجذْر

Une racine est purement consonantique, elle est formée par une suite de trois ou quatre (ou même cinq pour les noms /سَفَرْجَلْ/) consonnes formant la base du mot **[Abdelwahed, 1996]**. La racine est un élément important dans les langues dérivationnelle. En effet, à chaque racine correspond un champ sémantique et à l'aide de différents schèmes, on peut générer une famille de mots appartenant à ce champ sémantique, par exemple la racine «ك ت ب» [k t b] peut engendrer quinze mots autour de la notion de l'« écriture » tels que «كَاتِبٌ» [kâtibun] (écrivain), «مَكْتَبٌ» [maktabun] (bureau), «مَكْتَبَةٌ» [maktabatun] (bibliothèque) etc.

2. Le schème : الوَزْن

Le schème est un mot composé de trois consonnes ف [f], ع[‘], et ل [l], qui sont vocalisées et qui peuvent être augmentées par d'autres lettres (préfixe, suffixe et infixe). Le schème joue un rôle très important dans le processus de génération des formes dérivées à partir d'une racine. Ce processus de génération consiste à remplacer racine du schème par les consonnes de la racine en question, tout en gardant les mêmes voyelles et les mêmes lettres augmentées tout en respectant le même ordre des consonnes, autrement dit le schème peut être considéré comme une moule sur laquelle coule la racine (voir Figure12).

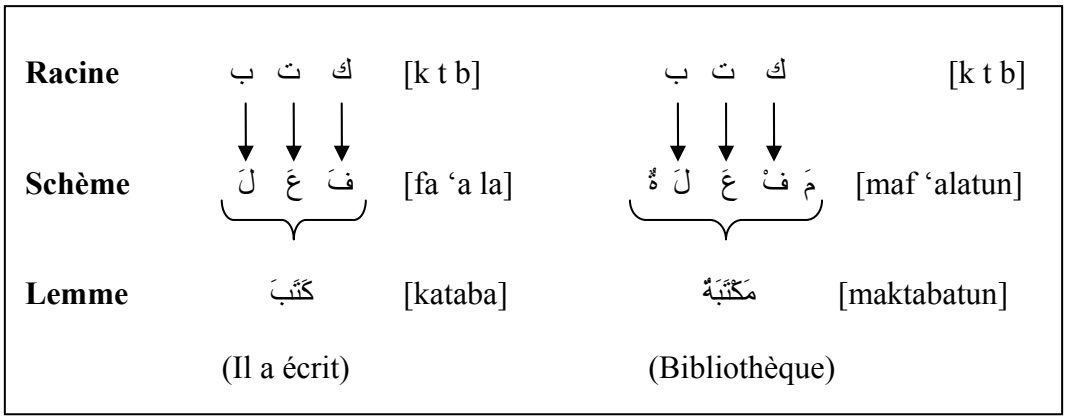


Figure 12 : Exemples de dérivation de la racine [k t b]

Dans cette figure, le lemme est formé par le remplacement respectif des consonnes du schème par les consonnes de la racine ك [k], ت [t], ب [b], tout en gardant les autres composants du schème.

On peut classer les schèmes en deux catégories : des schèmes verbaux et des schèmes nominaux. Ainsi, à partir d'une racine, on peut générer des noms et des verbes selon la catégorie du schème utilisé, par exemple à partir de la racine ك ت ب [k t b] on peut générer le verbe كَتَبَ [kataba] par le schème verbal فَعَلَ [fa'ala] et le nom مَكْتَبَةٌ [maktabatun] par le schème nominal مَفْعَلَةٌ [maf'alatun] (voir Figure 12).

3. Le lemme

Le lemme est l'entrée lexicale dans un lexique ou dans un dictionnaire. Il s'agit d'une forme entièrement vocalisée. Chaque mot est rapporté à son lemme qui est sa forme canonique qui dépend toujours de la catégorie grammaticale de ce mot : par exemple dans le Tableau 4, si c'est un nom il doit être au singulier et si c'est un verbe il doit être à l'accompli avec la troisième personne du singulier etc. Un lemme peut être formé par un mot simple ou un mot composé.

Catégorie grammaticale	Lemme	Mot
Nom	كِتَابٌ	كُتُبٌ
Verbe	كَتَبَ	كَتَبْتُ
Particule	عَلَى	عَلَى

Tableau 4 : Exemples de lemmes de catégories grammaticales différentes

Nous remarquons que les particules gardent toujours leur représentation de base. Pour les autres catégories le lemme permet de regrouper les mots ayant la même racine, le même schème original et le même sens. Ce regroupement aide à réduire le nombre d'entrées lexicales.

IV. Les catégories grammaticales

Selon la théorie grammaticale arabe ancienne, le lexique de la langue arabe comprend trois catégories de mots : verbe, nom et particule **[El-Dahdeh, 1996]**.

1. Verbe

Nous pouvons classer les verbes arabes selon plusieurs critères selon le nombre et la nature des consonnes de leurs racines et selon leurs schèmes aussi **[El-Dahdeh, 1999]**.

Selon le nombre des consonnes de la racine, nous avons soit des verbes trilitères qui ont trois consonnes, soit des verbes quadrilitères qui ont quatre consonnes peux nombreux.

Selon la nature des consonnes, nous avons soit des verbes sains (صحيح) qui ne sont pas formés par des lettres défectueuses[1], soit des verbes défectueux (معتلّ) qui contiennent une ou deux lettres défectueuses qui causent des altérations importantes au cours de la conjugaison. Un verbe peut contenir la lettre *hamza* ou *šadda* qui peuvent engendrer des conjugaisons irrégulières.

Selon le schème et le nombre de consonnes qui constituent la structure verbale, nous avons soit des verbes nus (مُجرّد) qui sont formés seulement par les consonnes de leurs racines et des voyelles brèves, soit des verbes augmentés ou dérivés (مزيد) qui sont dérivés de trois consonnes de la racine par modification des voyelles, par redoublement de la deuxième lettre de la racine, par adjonction et même par intercalation d'affixes. Les verbes dérivés se conjuguent avec les mêmes préfixes et suffixes que le verbe nu. Les verbes trilitères peuvent être augmenté au maximum par trois lettres et les verbes quadrilitères par deux lettres. Alors, la longueur maximale d'un verbe arabe est de 6 lettres.

2. Nom

Les noms arabes regroupent les substantifs, les adjectifs et les pronoms, ainsi que d'autre noms invariables **[Blachère et al, 1975]**. Les substantifs et les adjectifs sont créés en prenant pour origine tantôt un verbal tantôt un type nominal. Nous pouvons distinguer dans le Tableau 6 deux classes de noms : la première regroupe les noms conjugables ou semi conjugable qui peuvent avoir la forme duelle, plurielle etc. la deuxième classe regroupe les noms non conjugables qui gardent la forme quel que soit le contexte. Les noms conjugables sont soit des noms primitifs qui échappent à toute dérivation comme كَبْشٌ [kabšun] (bélier), soit des noms dérivationnels qui sont formés à partir d'une racine comme مَدْرَسَةٌ [madrasatun] (école) de la racine د ر س [d r s].

[1] Les lettres défectueuses sont (ا, و, ي)

Catégorie	Dérivation	Conjugaison	Sous-catégorie	Exemples
Nom	Dérivationnel irrégulier	Non conjugable	Adverbe	قَبْلَ، أَيْنَ، حَيْثُ
			Nom de voix	كِخْ، نِخْ
			Nom de verbe	هَيْهَاتَ، آهْ، أفٍّ
			Pronom Personnel (affixé ou isolé)	هُوَ، أنَا تُ، نُنَّ
			Pronom interrogatif	كَيْفَ، مَتَى، مَا
			Pronom conditionnel	مَنْ، إذَا
			Pronom allusif	كَمْ، كأيّ
		Conjugable	Pronom relatif	الذي، التي
			Nom de nombre	ثَلاثَةٌ، وَاحِدٌ، خَمْسَةٌ
			Pronom démonstratif	هَذَا،هَذِهِ
			Nom propre	مُحَمَّدٌ، هِنْدٌ، صَحْرَاءٌ
			Nom commun	قَلَمٌ، أرْنَبٌ، رَجُلٌ
	Dérivationnel régulier	Conjugable	Masdar	كِتَابَةٌ، القَتْلُ
			Participe actif	قَاتِل، شَارِب
			Participe passif	مَكْتُوب، مَضْرُوب
			Nom d'une fois	جَلْسَةٌ، ضَرْبَةٌ
			Nom de manière	نِظْرَة، جِلْسَةٌ
			Nom de temps	مَغْرِبٌ
			Nom de lieu	مَكْتَبٌ، مَقْبَرَةٌ
			Nom d'instrument	مِطْرَقَةٌ، مِسْمَارٌ
			Adjectif	حَسَنٌ، جَمِيلٌ، بَطَلٌ
			Elatif	أحْسَنُ، أفْظَلُ
			Nom diminutif	كُتَيِّبٌ، شُوَيْعِرٌ
			Nom de relation	تُونِسِيٌّ، مِصْرِيٌّ
			Intensif	قَتَّالٌ، غَوَّاصٌ

Tableau 5 : Classement des sous catégories de noms

3. Particule

Les particules sont des lemmes invariables et en nombre limité. Ils indiquent l'articulation de la phrase et ils servent à préciser les modalités des prépositions verbales et nominales **[El-Dahdeh, 1996], [Blachère et al, 1975]**. Malgré la difficulté de classer ces particules, on tenterait ce classement :

- ***Préposition :*** exemple (حَتَّى، عَنْ، ل، ك، ب)
- ***Particules de coordination :*** exemple (و، ف، ثُمَّ، أوْ)
- ***Particules interrogatives :*** exemple (مَا، هَلْ، أ)
- ***Particules d'affirmation :*** exemple (نَعَمْ، بَلَى، أجَلْ)
- ***Particules de négation :*** exemple (لَا، لَنْ، لَمْ)
- ***Particules distinctive :*** exemple (أيْ)
- ***Particules relatives :*** exemple (مَا)

- ***Particules de future :*** exemple (لَنْ، سَوْفَ، س)
- ***Particules conditionnelles :*** exemple (إنْ، لَوْ)

V. Les traits morphologiques

Les traits morphologiques arabes concernent la structure morphologique d'un mot, cela a un rapport avec les catégories grammaticales. Ce qui nous mène à citer les traits concernant les verbes et les traits concernant les noms, tout en éliminant les traits morphologiques des particules.

1. Les principaux traits du verbe

Un verbe arabe peut avoir six traits morphologiques :

1.1. L'aspect الصيغة

La conjugaison du verbe arabe est réduite par rapport aux langues indo-européennes. La notion de temps n'y a point de position solide, mais il y a la notion d'aspect du verbe **[Blachère et al, 1975]**. On en dénombre trois aspects :

- ***L'accompli* (المَاضِي)** : indique que l'action est achevée. C'est l'aspect le plus simple qui est utilisé avec la troisième personne du singulier pour représenter un verbe en remplacement de l'infinitif. Bien qu'il exprime le passé, il peut évoluer facilement au présent et au futur.
- ***L'inaccompli* (المُضَارَع)** : indique que l'action est en train de se réaliser, sans être accomplie. Il permet la modification des lettres principales du verbe. Il exprime le présent, et peut évoluer facilement au passé et au futur.
- ***L'impératif* (الأمْر)** : indique l'ordre ou la demande. Il peut être conjugué seulement avec les deuxièmes personnes. Généralement, il faut ajouter un *hamza* au début du verbe et terminer celui ci par la voyelle muette (ـْـ) **سُكُون [**sucûn**]**.

Nous pouvons mentionner que la détermination du temps, dans la langue arabe, ne se limite pas à l'analyse du verbe seulement, encore faut-il analyser toute la phrase.

1.2. Le mode

On parle de mode quand il s'agit de l'inaccompli, mais l'accompli et l'impératif, chacun d'eux a une seule modalité. L'inaccompli a trois modes qui diffèrent par leurs désinences **[Blachère et al, 1975]** :

- ***L'indicatif* (المَرفوع) :** employé dans une proposition principale ou isolée. Il se caractérise par une désinence (ـُـ) **ضَمَّة [**dammat**]** et par des flexions longues.

▪ ***Le subjonctif* (المَنصوب) :** utilisé en proposition subordonnée. Il se caractérise par une désinence (ـَـ) **فَتْحة** [fathat] et par des flexions courtes.

▪ ***L'apocopé* (المَجزوم) :** employé dans le conditionnel. Il se caractérise par l'absence de désinence (ـْـ) **سُكُون** [sucûn] et par des flexions courtes.

1.3. La voix

La langue arabe a deux voix :

▪***l'actif*** (المعلوم).

▪ ***Le passif*** (المجهول) : il a une signification et un emploi syntaxique assez différents de ceux du passif français. Pour des raisons sémantiques, seulement une partie des verbes admettant le passif se conjugue à toutes les personnes, en genre et en nombre **[El-Dahdeh, 1999]**.

1.4. La personne

Comme les autres langues, on en distingue trois :

▪ ***Première personne***: [naḥnu]نحن , ['anâ] أنا .

▪ ***Deuxième personne*** : ['anta] أنْتَ ، ['nti] أنْتِ، ['antumâ]أنْتُمَا، ['ntum] أنْتُمْ، ['atunna] أنْتُنَّ .

▪ ***Troisième personne*** : [huwa] هُوَ، [hiya] هِيَ، [humâ] هُمَا، [hum] هُمْ، [hunna] هُنَّ .

1.5. Le genre du verbe

Dans la langue arabe, il existe deux genres :

▪***Masculin***.

▪***Féminin.***

1.6. Le nombre du verbe

Un verbe arabe est pourvue de nombres : singulier, pluriel, et duel. **[Blachère et al, 1975]**.

▪ ***Le singulier***.

▪***Le duel***.

▪ ***Le pluriel***.

2. Les principaux traits du nom

Un nom arabe peut avoir cinq traits morphologiques :

2.1. Le genre du nom

Dans la langue arabe, il existe deux genres :

▪***Masculin*** : qui n'a pas d'indice ou représenté par un morphème zéro.

▪***Féminin*** : qui a la désinence ة، ات، اء، ى

Certains considèrent un troisième genre qui est le neutre.

2.2. Le nombre d'un nom

La langue arabe est pourvue de nombres : singulier, pluriel, et duel. Les grammairiens distinguent deux sortes de pluriels : le pluriel externe ou sain et le pluriel interne ou brisé **[Blachère et al, 1975]**.

- ***Le singulier*** : sa désinence dépend de sa flexion casuelle.
- ***Le duel*** : sa désinence dépend de sa flexion casuelle et de la catégorie grammaticale.
- ***Le pluriel*** : le pluriel externe ou sain est un pluriel à suffixe de masculin et de féminin.
- ***Le pluriel brisé*** : c'est le pluriel interne qui est utilisé avec les noms. C'est un pluriel qui n'a pas de désinence. Il a la même racine que le nom pris pour singulier, mais il est construit sur un autre schème qui est emprunté au vaste fond nominal comme مَدَارسُ [madârisu] (des écoles).

2.3. La flexion casuelle الإعراب

La flexion casuelle des noms est la terminaison qui est liée à leur catégorie dans la phrase. Elle peut être une voyelle brève ou une séquence de consonnes et de voyelles dans les cas du duel et du pluriel **[El-Dahdeh, 1999]**. Elle a trois cas :

- ***Nominatif*** : ***Sujet***
- ***Génitif : Complément direct***
- ***Accusatif*** : ***Complément indirect***

représentés dans le Tableau ci-dessous avec leurs désinences :

Nom	Singulier	Duel	Pluriel externe	
			Masculin	Féminin
Nominatif (Sujet)	ـُـ	ان	ون	اتُ
Génitif (Complément direct)	ـَـ	يْنِ	ينَ	اتِ
Accusatif (Complément indirect)	ـِـ	يْنِ	ينَ	اتِ

Tableau 6 : Les flexions casuelles d'un nom arabe

2.4. La détermination

C'est une question essentielle de savoir si un nom est déterminé ou indéterminé.

- ***Déterminé*** : il est signalé par une désinence vocalique sans *Tanwin*.
- ***Indéterminé*** : il est signalé par une désinence *Tanwin.*

Le nom peut être déterminé par :

1- le vocatif : يَا مُحَمَّدُ

2- l'annexion d'un complément de nom (بالإضافة): بَابُ الدَّارِ

3- l'article أَلْ : أَلْبَابُ

2.5. La définition

C'est une information de type booléen qui peut prendre les deux valeurs suivantes :

- ***Oui*** : si le nom est définissable par أَلْ comme بَابُ (أَلْبَابُ).
- ***Non*** : si le nom n'est pas définissable par أَلْ comme بَابٌ. Généralement, il s'agit des noms indéterminés par la désinence *Tanwin* ou les noms propres.

VI. Rôle du niveau phonologique dans la morphologie

L'application des règles phonologiques est un phénomène fréquent qui influence la morphologie d'un mot arabe. Ces règles sont liées surtout à la lettre *hamza*, aux lettres défectueuses et aux lettres dupliquées. Généralement, elles sont utilisées pour alléger la prononciation.

A titre d'exemple, nous présentons la règle phonologique spécifique pour les lettres défectueuses : "قلب حرف العلّة ألفا" **[Abdelwahed, 2002]** qui permet de remplacer une lettre défectueuse (و ou ي) par *alif* "ا", si la voyelle précédente est *fatha* (ـَـ) et sa voyelle n'est pas *sukûn* (ـْـ). Cette règle est illustrée par les exemples de la Figure 13 dont le premier montre les conditions d'appliquer cette règle à la forme abstraite générée par le mécanisme de dérivation. Dans le deuxième exemple, la voyelle de la lettre défectueuse est *sukûn*, alors on n'applique pas la règle car ses conditions ne sont pas vérifiées.

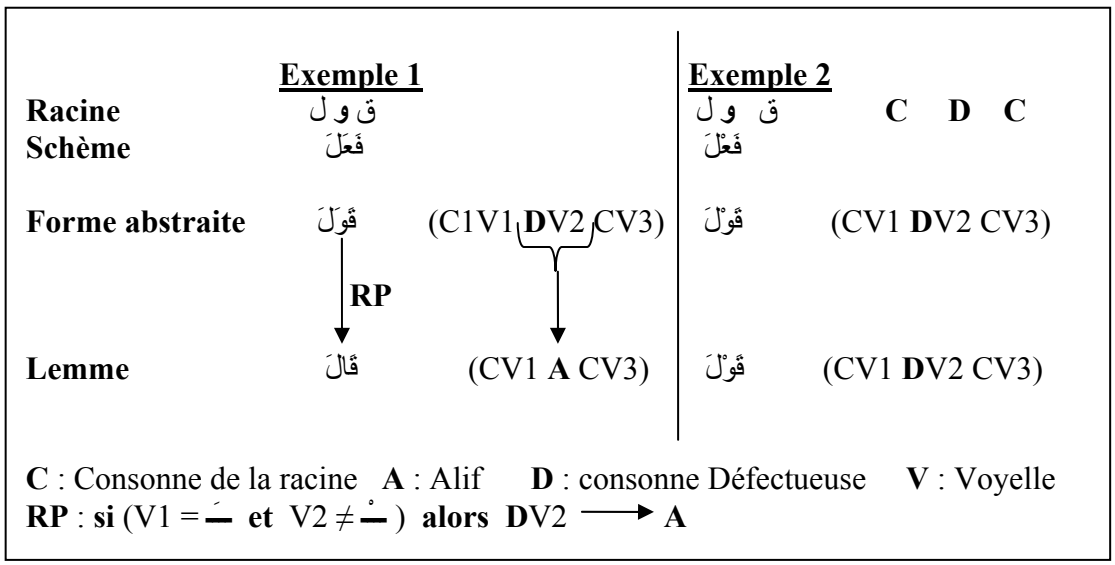


Figure 13 : Application d'une règle phonologique

VII. Conclusion

La langue arabe possède ses propres caractéristiques qui sont différentes par rapport aux langues indo-européennes. Elle se distingue par le lien étroit entre ses différents niveaux linguistiques : phonologique, morphologique, syntaxique et sémantique. Dans ce chapitre, nous avons étudié certaines caractéristiques de la langue arabe, notamment celle d'ordre phono-morphologique. Il serait intéressant d'étudier l'application des propositions de la norme LMF sur le cas de la langue arabe.

CHAPITRE

3

Application de LMF à la langue arabe

I. Introduction

La proposition de la norme LMF pour les bases lexicales permet de promouvoir l'interopérabilité entre des bases de langues différentes. Dans ce cadre, des études avancées ont été élaborées sur plusieurs langues indo-européennes, en vue de tester leurs conformités par rapport aux propositions de LMF. De notre part, nous avons pris l'initiative d'appliquer cette norme à la langue arabe, d'une part, pour enrichir la norme par les exigences de cette langue, et d'autre part, pour élaborer une base lexicale arabe normalisée pour les applications de TALN.

Nous rappelons que le méta-modèle de LMF comporte un noyau et cinq extensions pour le TALN. La flexibilité de ces propositions nous permet de sélectionner seulement une partie du méta-modèle. Cette partie peut être étendue à tout moment par les autres parties restantes. Ainsi, nous allons nous limiter aux extensions nécessaires à la morphologie en vue de les appliquer sur la langue arabe. Pour ce faire, nous allons choisir des supports susceptibles de réaliser une étude qui nous conduit aux solutions envisagées.

Nous allons commencer par préciser la démarche de réalisation à suivre. Ensuite, nous indiquerons les catégories de données nécessaires, puis, nous présenterons les modélisations du noyau, de l'extension morphologique et de l'extension de modes de flexions. Nous terminons par donner certaines critiques et propositions à l'issue de notre investigation.

II. Démarche à suivre

Un lexique conforme à LMF est défini comme une combinaison d'un noyau et d'une ou de plusieurs extensions lexicales avec un ensemble de catégories de données. Cette combinaison est décrite dans le diagramme d'activité UML dans la Figure 14.

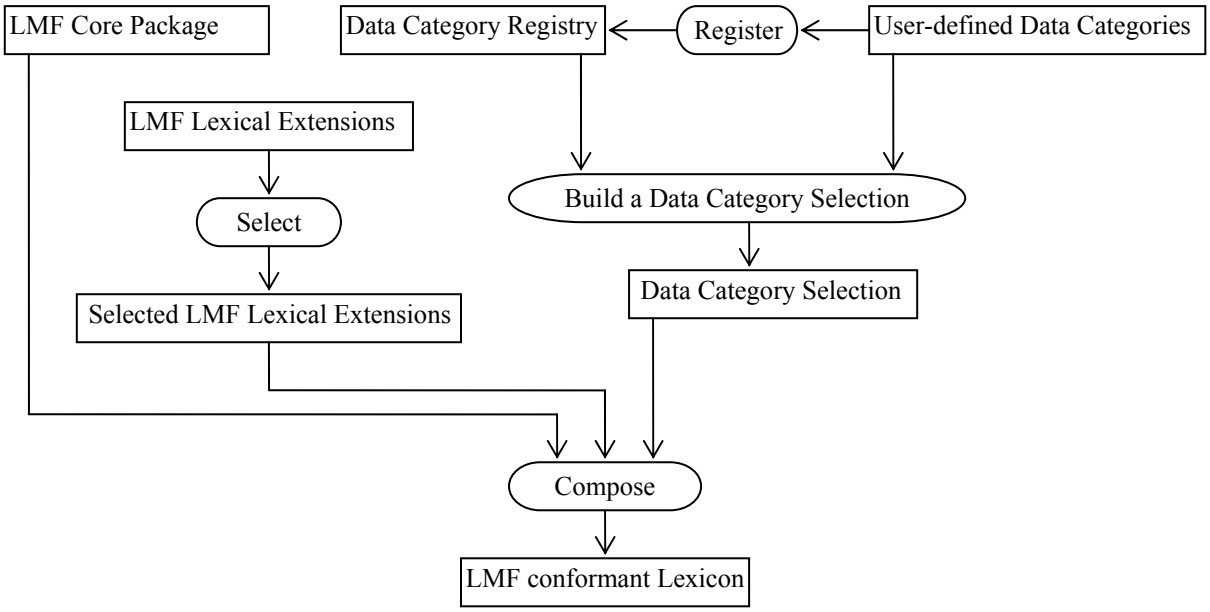


Figure 14 : Démarche de réalisation

Dans cette démarche, nous allons préparer le noyau et sélectionner les sous-ensembles des extensions : morphologique et de modes de flexion proposés dans la norme. En parallèle, nous allons sélectionner les catégories de données nécessaires pour la langue arabe à partir du RCD. Au cas où une information linguistique nécessaire pour ces deux extensions ne serait pas représentée dans ce registre, nous devons définir une nouvelle catégorie de données, en respectant la norme ISO 12620, qui sera utilisée dans nos extensions et ajoutée dans le RCD après l'autorisation des responsables de cette norme.

Nous rappelons que LMF a deux représentations concurrentes : l'une extensionnelle (les formes fléchies) et l'autre intentionnelle (les paradigmes de flexions). Nous allons utiliser la première pour représenter tous les cas particuliers ou les entrées qui n'ont pas de paradigmes actuellement. Nous profitons de la deuxième représentation pour réduire la taille de notre base lexicale en utilisant les paradigmes de flexions.

III. Sélection des catégories de données

Nous avons déjà présenté la partie conceptuelle d'une catégorie de données et les traits d'un profil morpho-syntaxique. Cependant, la sous-catégorisation des catégories de données n'est pas valable pour la langue arabe. Ce qui nous amène à proposer de considérer la sous-catégorisation comme une information spécifique pour chaque langue et n'est pas comme une information générale, cette proposition sera détaillée dans le paragraphe VII.

Actuellement, nous allons sélectionner les catégories de données à partir du RCD, ensuite nous ajoutons les catégories de données manquantes qui seront représentées dans l'annexe A, en respectant le nouvel modèle proposé.

1. Les catégories de données normalisées

Dans le Tableau 7, nous allons présenter les catégories de données : complexes et simples. Pour les catégories de données complexes, s'il y a des valeurs manquantes, nous les représenterons, dans le tableau, soulignées et nous allons les détailler dans le paragraphe suivant.

Catégorie de données complexe	Catégorie de données simple	Commentaire
case	nominativeCase genetiveCase accusativeCase	C'est un trait morphologique utilisé au niveau d'*InflctedForm.* généralement il lié à la terminaison du mot.
definiteness	Definite Indefinite	C'est un trait morphologique utilisé au niveau d'*InflctedForm* pour les noms.
grammaticalGender	Masculine Feminine Neutre	C'est un trait morphologique utilisé au niveau d'*InflctedForm* pour les verbes et les noms (ou au niveau de *LemmatisedForm* pour quelques noms).
grammaticalNumber	Singular Dual Plural Plural broker	C'est un trait morphologique utilisé au niveau d'*InflctedForm* pour les verbes et les noms.
voice	activeVoice passiveVoice	C'est un trait morphologique utilisé au niveau d'*InflctedForm* pour les verbes.
person	firstPerson secondPerson thirdPerson	Idem à voice.
verbFormMood	Indicative	C'est un trait morphologique

	Subjonctive Apocopate	utilisé au niveau d'*InflctedForm* pour les verbes et seulement lorsque l'aspect est inaccompli.
writtenForm		C'est l'orthographe d'un lemme (*LemmatisedForm*) ou une forme fléchie (*InflctedForm*).
partOfSpeech	/adjective/ /adposition/ /adverb/ /affirmativeParticle/ /bullet/ /closeParenthesis/ /colon/ /comma/ /commonNoun/ /copula/ /definiteArticle/ /demonstrativePronoun/ /exclamativePoint/ /exclamativePronoun/ /futureParticle/ /interjection/ /interrogativeParticle/ /interrogativePoint/ /interrogativePronoun/ /invertedComma/ /negativeParticle/ /noun/ /openParenthesis/ /particle/ /pastParticipleAdjective/ /personalPronoun/ /point/ /preposition/ /presentParticipleAdjective/ /pronoun/ /properNoun/ /punctuation//relativePronoun/ /semiColon/ /slash/ /suspensionPoints/ /verb/	C'est la catégorie grammaticale liée à chaque *Lexical Entry*.

Tableau 7 : Les catégories de données normalisées

2. Les nouvelles catégories de données

Les catégories de données présentées dans le Tableau 8 sont détaillées Annexe B. Néanmoins, elles ne sont pas encore publiées dans la norme ISO 12 620.

Catégorie de données complexe	Catégorie de données simple	Commentaire
elInclusion	no yes	C'est un trait morphologique qui prend la valeur *yes* si le nom peut être défini par "الْ" et *no* sinon.
grammaticalNumber	pluralBroker	
havePassive	no/yes	
verbFormMood	apocopate	
verbFormAspect	Accomplished Unaccomplished Imperative	L'accompli a une seule modalité et l'inaccompli a trois modalités.
partOfSpeech	numberNoun voiceNoun affixedPersonalPronoun conditionalPronoun allusivePronoun Masdar onceNoun mannerNoun timeNoun placeNoun instrumentNoun elative diminutiveNoun relationNoun intensive coordinationParticle distinctiveParticle relativeParticle conditionalParticle	

root		La racine arabe comporte seulement des consonnes séparées par des espaces.
scheme		Le schème arabe est de racine [ف ع ل], totalement voyellé.

Tableau 8 : Les catégories de données non normalisées

Les catégories de données soulignées, dans le tableau ci-dessus, sont déjà normalisées.

IV. Modélisation du noyau

Dans la modélisation du noyau, il y a six classes qui nous intéressent, à savoir : *Database*, *Lexicon*, *Lexicon Information*, *Lexical Entry*, et les deux classes de spécification de *Form* : *LemmatisedForm* et *inflectedForm.*

Le squelette structurel du noyau a pour racine *Database*, contenant l'attribut : dtdVersion qui prend la valeur "1.0". Cette racine peut avoir plusieurs catégories de données qui représentent des informations administratives comme la date d'une version. Une *Database* peut contenir un ou plusieurs *Lexicon* auquel nous attachons toutes sortes d'informations dans un objet *Lexicon Information.*

Un *lexicon* est composé par plusieurs *LexicalEntry* qui portent un identifiant et l'information de catégorie grammaticale. Ce choix implique la création de deux entrées pour des formes identiques à catégorie grammaticale distincte (كَمْ : pronom interrogatif vs. كَمْ : pronom allusif (كِنَايَة)). En contrepartie, des homonymes à catégorie grammaticale identique (سَاعَةٌ : une heure vs. سَاعَةٌ : une montre) ne sont pas traités dans deux entrées différentes.

Pour la langue arabe la catégorie verbe et la plupart des sous catégories de nom sont identifiées par une racine et un schème. En effet, nous ajoutons /root/ et /scheme/ comme des catégories de données dans la classe *LexicalEntry* et nous ne pouvons pas les mettre dans la classe *LemmatisedForm* car la combinaison d'une racine et d'un schème peut générer deux lemmes différents comme la racine "ط ي ر" [ṭ y r] avec le schème "تَفَعَّلَ" [tafa‘‘ala] donne deux lemmes différents l'un est le résultat de mécanisme de dérivation seulement et l'autre est le résultat de mécanisme de dérivation et une règle phonologique pour alléger sa prononciation**,** comme le montre la Figure 16 suivante.

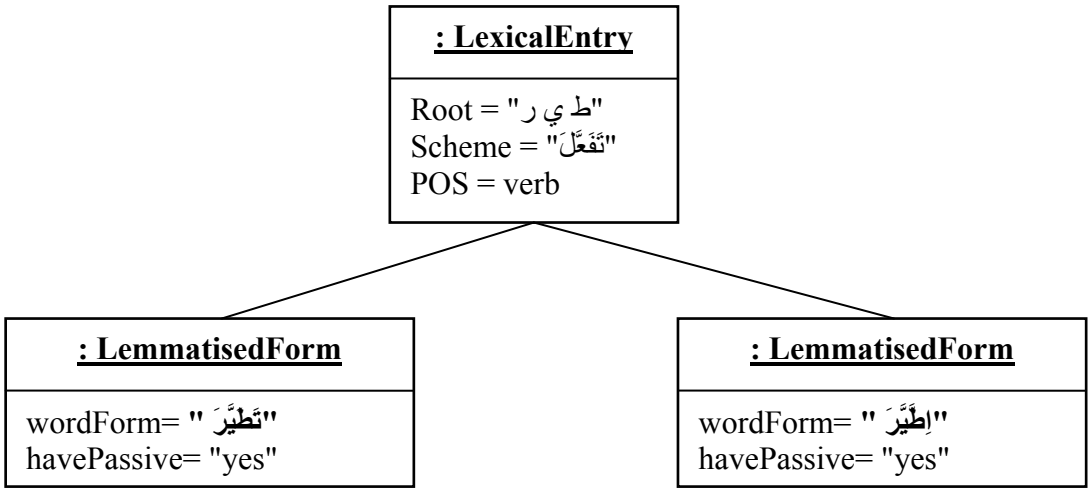


Figure 15 : Exemple de la racine "ط ي ر " avec le schème "تَفَعَّلَ "

Nous remarquons que la cardinalité 0..* entre *LexicalEntry* et *LemmatisedForm* est intéressante pour la langue arabe lorsqu'il s'agit de résoudre des problèmes comme l'exemple ci-dessus (Figure 16). Cette représentation nous permet d'associer à chaque lemme ses formes fléchies et avec les catégories de données : /wordForm/ pour l'orthographe du lemme et /havePassive/[2] qui est nécessaire seulement pour la catégorie grammaticale verbe.

En plus, ce choix implique la création de deux entrées pour des formes identiques à schèmes distincts qui donnent deux sens différents.

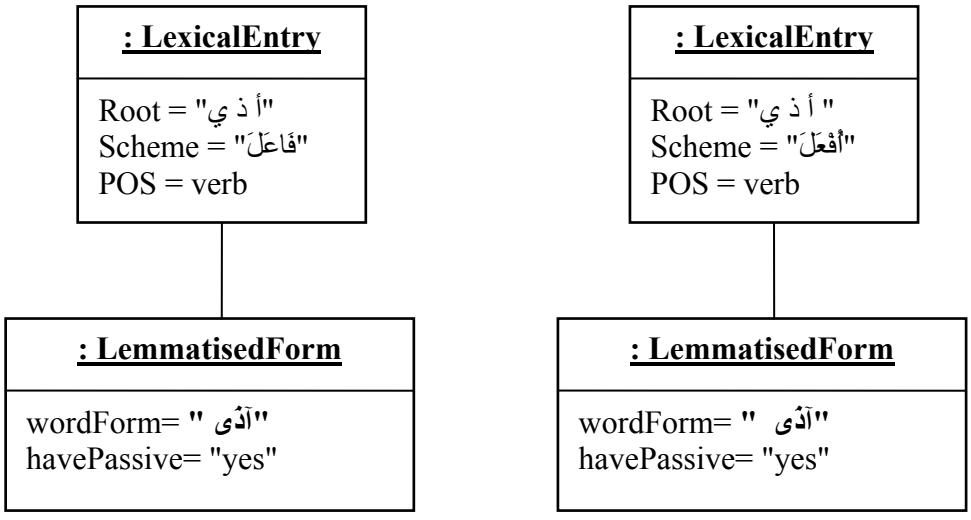


Figure 16 : Exemple de "آذَى "

On a appliqué le mécanisme de dérivation sur les deux entrées lexicales présentées dans la Figure 17, ensuite on a utilisé deux règles phonologiques différentes pour alléger la

[2] Cette catégorie est définie au niveau des paradigmes de flexion dans le paragraphe suivant, qui prend **no** si le verbe admet seulement la voix active et **yes** si le verbe admet les deux voix.

prononciation : la première consiste à fusionner la *hamza* de la racine et l'*alif* du schème, et la deuxième fusionne la *hamza* de la racine et la *hamza* du schème.

V. Modélisation de l'extension Morphologique

Cette extension est traitée de deux manières différentes dans LMF. La première présente l'ensemble des formes fléchies d'un lemme. La deuxième fait référence à un paradigme de flexion qui doit être détaillé dans l'extension des modes de flexion et qui va faciliter la génération des formes fléchies. Par conséquent, les entrées lexicales invariables (adverbe, particule…) n'ont pas de paradigmes de flexion.

Une Forme fléchie *inflectedForm* correspond à une forme d'occurrence du lemme qui doit avoir une forme fléchie même pour les invariables. Ce choix permet d'unifier la représentation des entrées (même structure, indépendamment de la catégorie grammaticale) et la maintenance de la base.

1. Cas des verbes

Actuellement, nous allons suivre la représentation intentionnelle en utilisant les paradigmes de flexion pour réduire la taille de la base. Nous allons également élaborer un programme permettant la génération des formes fléchies, à partir du lemme, temporellement, pour faciliter l'exploitation de la base.

Pour une forme fléchie ou une combinaison de traits morphologiques d'un verbe, on associe les catégories de données suivantes :

/grammaticalGender/ : genre (seulement avec 2ème et 3ème personne).

/grammaticalNumber/ : nombre.

/person/ : personne grammaticale.

/voice/ : voix.

/verbFormAspect/ : aspect grammatical.

/verbFormMood/ : mode grammatical (seulement avec l'inaccompli).

Dans la Figure 18, nous donnons les deux représentations : intentionnelle qui fait référence au paradigme convenable pour le verbe "**كَتَبَ** " et extensionnelle qui représente un échantillon des formes fléchies de ce verbe tels que "**كَتَبْنَ**", "**أَكْتُبُ**", "**يُكْتَبْ**" et "**أُكْتُبُوا**" avec leurs différents traits morphologiques.

Nous remarquons que le nombre des traits morphologiques peut varier d'une forme fléchie à une autre parce qu'il y a des traits morphologiques que leur présence dépend d'un autre trait : par exemple le genre est absent avec la première personne (firstPerson).

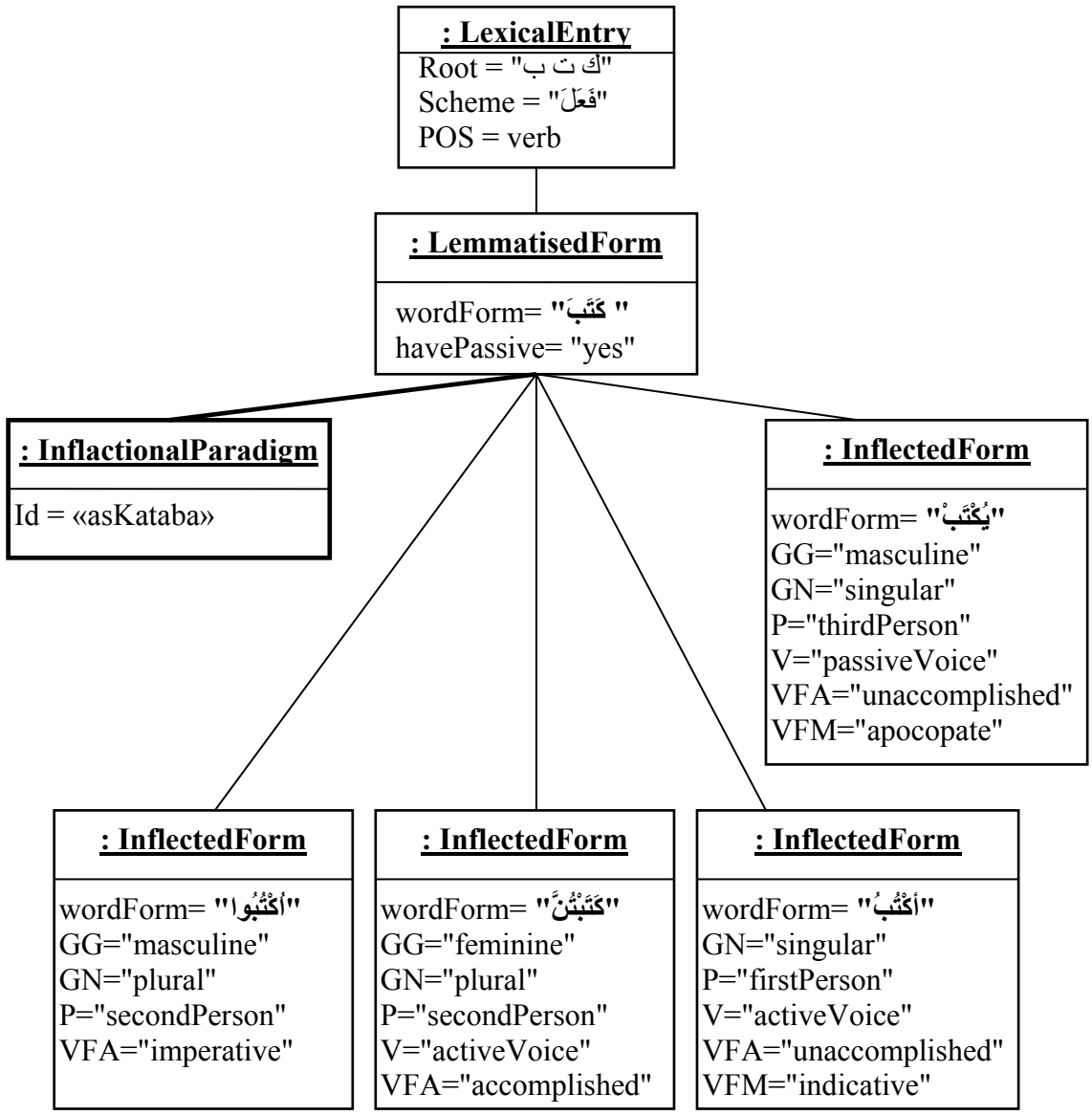


Figure 17 : Exemple d'une entrée lexicale verbale

Généralement pour chaque verbe, nous spécifions ses paradigmes de flexion. Sachant qu'un verbe peut avoir plusieurs lemmes et que chacun d'eux est lié à un paradigme : le lemme تَطَيَّرَ [taṭayyara] qui est lié au paradigme " asTafaqqada"[3] et le lemme اِطَّيَّرَ [iṭṭayyara] lié au paradigme "asIssamma'a"[4] en sont des exemples édifiants présentés dans la Figure 16.

[3] C'est un identifiant d'un paradigme qui a comme verbe type "تَفَقَّدَ"
[4] C'est un identifiant d'un paradigme qui a comme verbe type "اِسَّمَّعَ"

2. Cas des noms

Les noms ont plusieurs sous-catégories qui peuvent être variables ou invariables. Généralement les noms variables ont plusieurs formes fléchies qui sont associées aux catégories de données suivantes :

/writtenForm/ : orthographe de la forme fléchie.

/grammaticalGender/ : genre : le nom, quand il garde toujours le même genre, celui-ci sera associé au lemme.

/grammaticalNumber/ : nombre.

/flexionCasuelle/ : flexion casuelle.

/definiteness/ : défini ou non défini.

/elInclusion/ : peut être défini par "الْ".

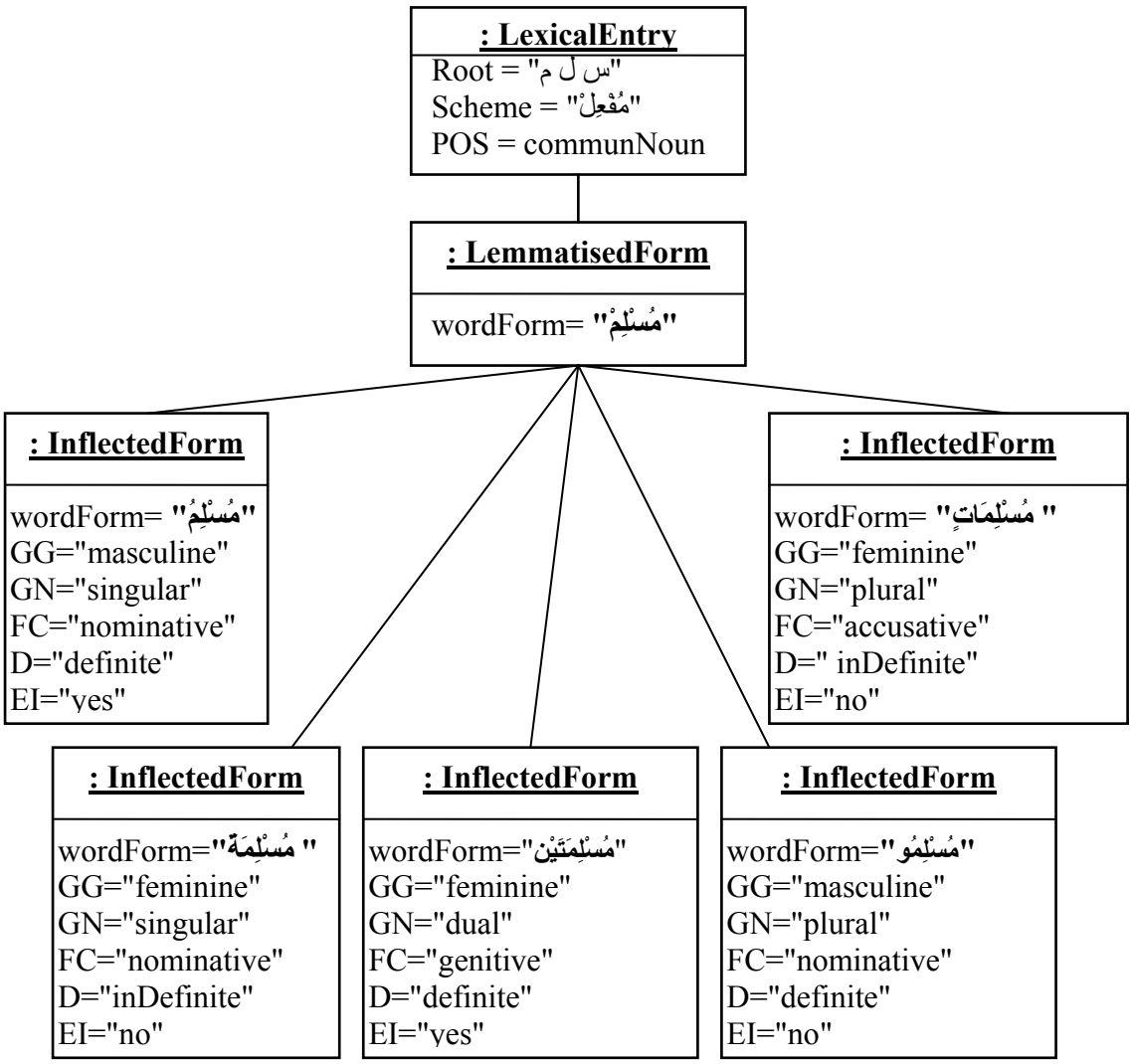


Figure 18 : Exemple d'une entrée lexicale nominale

Dans la Figure 19, nous donnons la représentation extensionnelle pour le nom "**مُسْلِمْ** ", en proposant, un échantillon des formes fléchies de ce nom tels que "**مُسْلِمُ** ", "**مُسْلِمَةٌ** ", "**مُسْلِمَتَيْنِ**", "**مُسْلِمُو** " et "**مُسْلِمَاتٍ** " avec leurs différents traits morphologiques.

En arabe, le nom peut avoir 36 formes fléchies comme le montre le Tableau 9, mais si son genre est fixe, il n'en aura que 18 seulement, autrement dit, la moitié.

Si un nom est au duel féminin ou masculin ou au pluriel masculin, il peut être défini soit avec l'article **ال**, soit par un syntagme nominal (بالإضَافة). Dans le deuxième cas il ne pourra pas admettre **ال** et se verra supprimé le ن à la fin**.**

		Masculin			Féminin		
		Normatif	**Génitif**	**Accusatif**	**Normatif**	**Génitif**	**Accusatif**
Singulier	**Défini** ال +	مُسْلِمُ	مُسْلِمَ	مُسْلِمِ	مُسْلِمَةُ	مُسْلِمَةَ	مُسْلِمَةِ
	Indéfini ال -	مُسْلِمٌ	مُسْلِمًا	مُسْلِمٍ	مُسْلِمَةٌ	مُسْلِمَةً	مُسْلِمَةٍ
Duel	ال +	مُسْلِمَانِ	مُسْلِمَيْنِ	مُسْلِمَيْنِ	مُسْلِمَتَانِ	مُسْلِمَتَيْنِ	مُسْلِمَتَيْنِ
	Défini ال -	مُسْلِمَا	مُسْلِمَيْ	مُسْلِمَيْ	مُسْلِمَتَا	مُسْلِمَتَيْ	مُسْلِمَتَيْ
Pluriel	ال +	مُسْلِمُونَ	مُسْلِمِينَ	مُسْلِمِينَ	مُسْلِمَاتُ	مُسْلِمَاتِ	مُسْلِمَاتِ
	Défini ال -	مُسْلِمُو	مُسْلِمِي	مُسْلِمِي	مُسْلِمَاتٌ	مُسْلِمَاتٍ	مُسْلِمَاتٍ

Tableau 9 : Exemple des formes fléchies d'un nom

Nous remarquons qu'un nom peut avoir soit le pluriel régulier, soit un ou plusieurs pluriels brisés tels que [kilâbun], ['aklubun]. Un pluriel brisé est traité comme étant une forme au singulier et aura en conséquence ses mêmes flexions casuelles (du défini et de l'indéfini).

La préparation des paradigmes de flexion pour les noms est un projet en cours de discussion. Cependant, nous allons suivre la représentation extensionnelle.

3. Cas des particules

Ce sont des entrées lexicales invariables, autrement dit, ils n'ont ni formes fléchies ni paradigmes. Leur rôle se limite principalement au niveau syntaxique et leur influence est quasi nulle sur le plan morphologique. La Figure 20 en fournit un exemple.

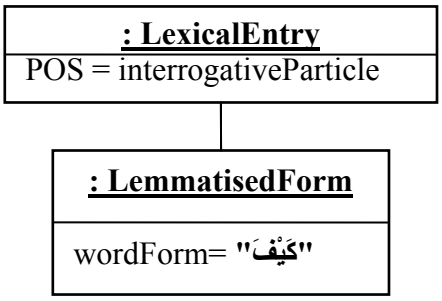


Figure 19 : Exemple d'une entrée lexicale particule

VI. Modélisation de l'extension des modes de flexion : les paradigmes de flexion

Les paradigmes de flexion sont des structures à un grand nombre de mots permettant une description intentionnelle. Cette extension est intéressante pour les langues complexes (la conjugaison génère plusieurs formes fléchies) comme la langue arabe qui a plus que 16 000 verbes et qui est particulièrement l'objet de notre étude. Selon Ammar, un verbe peut avoir 109 formes fléchies quand il admet la voix active et passive et 57 formes fléchies seulement lorsqu'il n'admet que la voix active **[Ammar et al, 1999]**.

Notons que en absence d'études suffisantes sur les flexions des noms arabes, nous allons se limiter dans cette modélisation aux cas des verbes. Dans la langue arabe la voix, qu'elle soit passive ou active, influence le calcul des formes fléchies d'un verbe, il y a alors des verbes qui admettent la voix passive et d'autres non. Cependant, d'après la norme, toutes les *MorphologicalFeaturesCombinator* d'un paradigme sont appliquées sur le lemme pour générer toutes les formes fléchies. Ainsi, nous avons deux solutions :

- Soit nous créons deux paradigmes, l'un avec les deux voix, active et passive, et l'autre avec la voix active seulement.
- Soit nous ajoutons une catégorie de données au niveau du lemme qui prend la valeur "no" si le verbe admet seulement la voix active et la valeur "yes" si le verbe admet les deux voix.

Nous remarquons que l'application de la première solution engendre le redoublement des paradigmes, alors que la deuxième solution est plus intéressante et évite la répétition des paradigmes de la voix active.

Le principe de conjugaison des verbes arabes est simple, mais l'application de celui-ci nécessite l'intervention des règles générales d'ordre phonologique, ce qui va compliquer la tâche. En effet, nous allons essayer d'appliquer cette extension à la conjugaison des verbes arabes, en précisant les cas particuliers qui nécessitent l'application d'une règle phonologique.

1. Combinaison des traits morphologiques

Un paradigme comprend plusieurs *MorphologicalFeaturesCombinator* qui est une classe abstraite. Cette classe relie un ensemble de *MorphologicalFeature* qui varie selon la catégorie grammaticale, avec un ou plusieurs *InflectedFormCalculator,* ce choix nous permet de générer plus qu'une forme fléchie pour une seule combinaison de traits morphologiques comme l'exemple de **شَدَّ** [šadda] avec l'inaccompli apocopé et avec le pronom personnel **هُو** et qui comporte les deux formes **يَشْدُدْ** [yašdud] ou **يَشُدَّ** [yašudda] présentées dans la Figure 21**.**

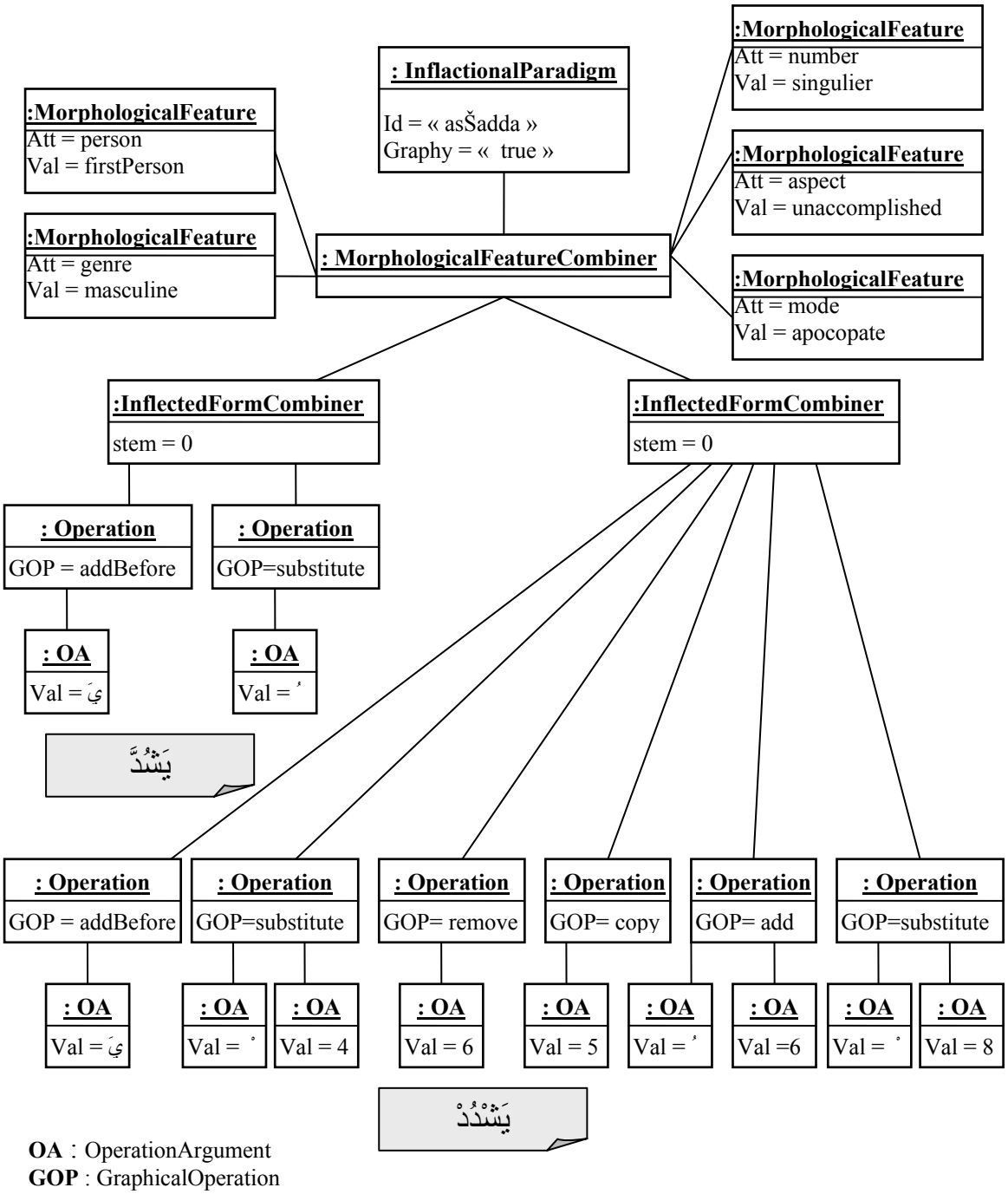


Figure 20 : Exemple de calcul de deux formes fléchies ayant les mêmes traits morphologiques

2. Calcul d'une forme fléchie

Un objet *InflectedFormCalculator* est composé par des opérations (/add before/, /add after/, /substitute/, /copy/, /add/, /remove/, /remove after/) dont l'utilisation sera fixée selon un ordre bien déterminé. Nous allons utiliser ces opérations pour appliquer les règles de conjugaison de l'arabe et nous allons essayer de limiter et de traiter les cas particuliers qui peuvent être apparus après l'application de ces règles. Autrement dit, lors

de l'application des règles de conjugaison, il y a des règles phonologiques qui doivent être appliquées.

Dans le cas le plus simple, c'est-à-dire là où il n'y a pas de déclenchement d'une règle phonologique, le calcul d'une forme fléchie peut utiliser : le changement des voyelles, l'ajout d'un préfixe ou d'un suffixe, et la suppression d'un *hamza wasliya* au début ou une voyelle à la fin.

2.1. Transformation des voyelles

La transformation des voyelles est un phénomène fréquent qui peut toucher la plupart des voyelles du lemme lors du calcul d'une forme fléchie comme dans l'exemple de la Figure 22.

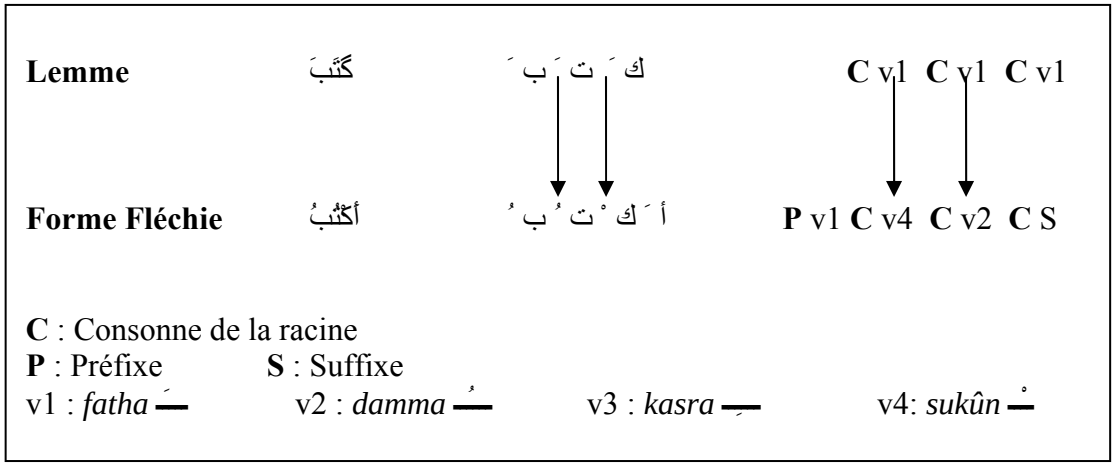


Figure 21 : Exemple de calcul de la forme fléchie "أَكْتُبُ "

D'après l'exemple de la Figure 22, il est bien clair qu'il faut modifier les deux voyelles du lemme pour trouver la forme fléchie. Dans LMF, il y a l'opération *substitute* qui permet de remplacer une chaîne de caractère par une autre. Celle-ci nous permet de remplacer une voyelle par une autre tout en précisant la position de l'opération et la nouvelle voyelle.

2.2. Ajout d'un préfixe ou d'un suffixe

L'ajout d'un préfixe ou d'un suffixe dépend des traits morphologiques de la forme fléchie en question. En plus, un suffixe peut être une voyelle comme dans l'exemple précédent ou un pronom personnel affixé comme dans l'exemple de la Figure 23.

Lemme	كَتَبَ	**C**v**C**v**C**v
Forme Fléchie	يَكْتُبْنَّ	**P(**يَ**)** + **C**v**C**v**C**v + **S(**نّ**)**
C : Consonne de la racine **P** : Préfixe	**S** : Suffixe	v : voyelle

Figure 22 : Calcul d'une forme fléchie "يَكْتُبْنَّ "

Pour ajouter un préfixe et un suffixe, nous utilisons respectivement les opérations « add Before » et « add after ».

2.3. Suppression d'un *hamza wasliya* ou de la dernière voyelle

La suppression de la dernière voyelle est fréquente et elle est toujours suivie par l'ajout d'un suffixe. Mais la suppression de *hamza wasliya* est appliquée seulement avec les verbes augments par "اِسْتَ" et elle est toujours suivie par l'ajout d'un préfixe tel que l'exemple dans la Figure 24 :

Lemme	اِسْتَقْبَلَ	(0)
	سْتَقْبَلَ	(1) suppression de *hamza wasliya*
	يَسْتَقْبَلَ	(2) ajout d'un préfixe
	يَسْتَقْبِلَ	(3) changement d'une voyelle
	يَسْتَقْبِل	(4) suppression de la dernière voyelle
Forme Fléchie	يَسْتَقْبِلُونَ	(5) ajout d'un suffixe

Figure 23 : Calcul de la forme fléchie "يَسْتَقْبِلُونَ "

Dans cet exemple, nous remarquons que les étapes (1) et (4) représentent des préparations respectivement pour les étapes (2) et (5).

3. Les cas liées à la phonologie

La complexité morphologique de la langue arabe se représente dans l'utilisation des règles phonologiques au cours du calcul d'une forme fléchie. Pour éviter l'apparition des formes erronées, nous limitons les altérations qui peuvent être induites par la lettre *hamza,* les lettres défectueuses ou le signe *šadda* au cours de la conjugaison. En plus, nous avons besoin de fusionner la première consonne du suffixe avec la dernière consonne de la racine lorsqu'elles sont identiques.

3.1. Transformation de la graphie de *hamza*

En général, la graphie de *hamza* dépend de sa position, de sa voyelle et de la voyelle précédente. Nous avons déjà dit que le calcul des formes fléchies nécessite la transformation des voyelles. En effet, ces changements peuvent être avant ou après la

lettre *hamza.* Dans ce cas, il faut suivre les règles phonologiques et transformer la graphie de cette lettre. Par exemple, si nous avons changé la voyelle du *hamza* de *fatha* ـَ à *kasra* ـِ alors cette lettre sera sur ـِئ qui donne la graphie suivante ئِ.

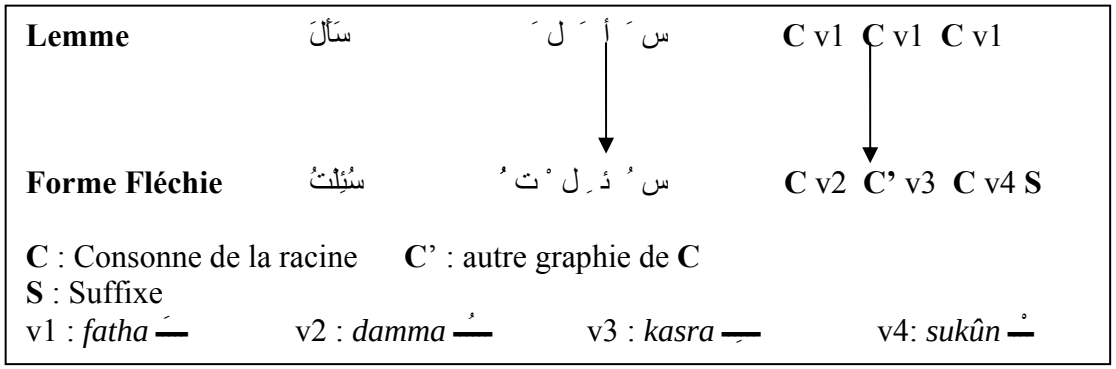


Figure 24 : Exemple de transformation d'un hamza

Dans le cas de la Figure 25 , nous utilisons l'opération « *substitute* » pour remplacer la graphie d'une consonne par une autre, en précisant toujours la position de l'opération et la nouvelle graphie.

3.2. Transformation des lettres défectueuses

La présence des lettres défectueuses dans la racine peut causer des altérations importantes au cours de la conjugaison. Sachant que ces lettres peuvent figurer sous forme de *alif* dans le lemme, alors que lors de la conjugaison elles reprennent leur forme originale. L'origine de ces altérations est le changement des voyelles qui nécessite l'application d'une règle phonologique. Par exemple, la racine "ق و ل" [q w l] avec le schème "فَعَلَ" [fa'ala] et selon la règle phonologique donne le lemme "قَالَ" [qâla] et pendant le calcul de la forme fléchie à l'inaccompli actif, on doit remplacer *alif* ا par la forme originale de la racine : و car la voyelle précédente *fatha* ـَ est remplacée par *damma* ـُ.

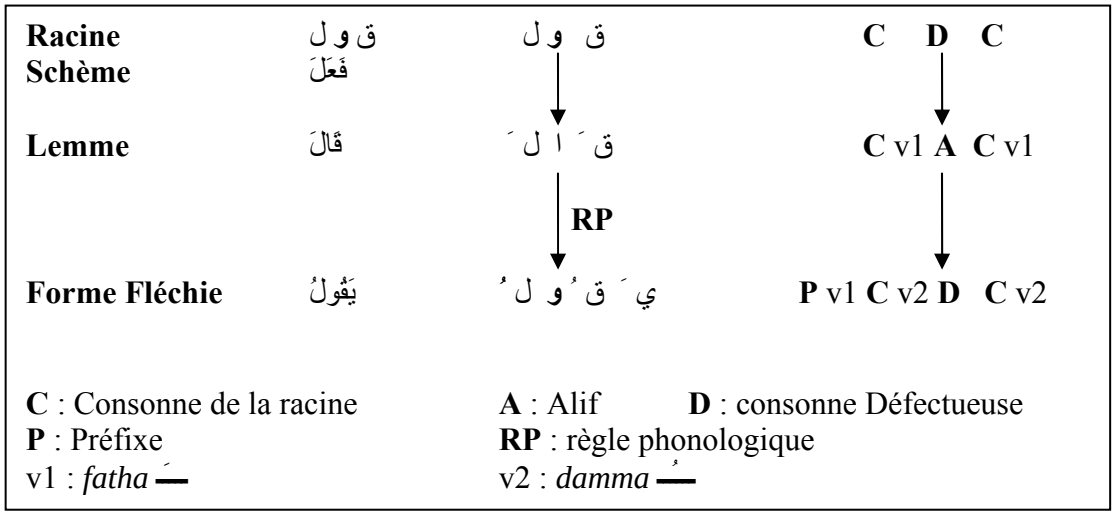


Figure 25 : Exemple de transformation d'une lettre défectueuse

Dans l'exemple de la Figure 26, nous utilisons l'opération « *substitute* » pour remplacer une consonne défectueuse par *alif*, en précisant à chaque fois la position de l'opération et la nouvelle lettre.

3.3. Transformation de *šadda*

Parmi les phénomènes qui nécessitent un traitement particulier, nous trouvons le signe *šadda* qui peut être figuré dans les lemmes des verbes dupliqués (exemple : مَدَّ [madda]). La consonne surmontée de ce signe peut être remplacée par deux consonnes identiques au cours de la conjugaison.

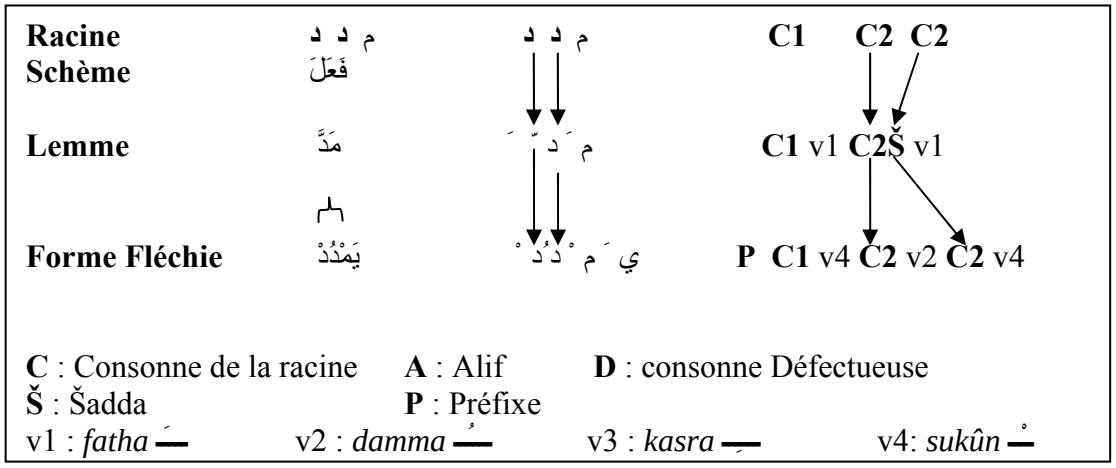


Figure 26 : Exemple de transformation de *šadda* pour le cas de "مَدَّ "

Dans le cas de la Figure 27, nous calculons la forme fléchie à partir du lemme par l'ensemble des opérations suivantes exécutées selon l'ordre suivant :

-1-« addBefore » : pour ajouter le préfixe يَ.

-2 -« substitute » : pour remplacer la voyelle à la position 4.

-3-« remove » : pour supprimer *šadda* à la position 6.

-4-« copy » : pour dupliquer la consonne à la position 5.

-5-« add » : pour ajouter la voyelle à la position 6.

-6 -« substitute » : pour remplacer la voyelle à la position 8.

Les opérations de 3 à 6 sont spécifiques pour transformer le signe de *šadda* par une consonne.

3.4. Fusion du suffixe avec la dernière consonne de la racine ن ou ت

Au cours du calcul d'une forme fléchie, l'ajout d'un suffixe peut donner des formes erronées qui sont rencontrées avec les racines qui se terminent soit par la lettre ن, soit par la lettre ت.

La présence de la lettre ن à la fin de la racine, engendre la fusion de celle-ci avec le suffixe ن du féminin pluriel tel que حَزَنَّ [hazanna] présenté dans la Figure 28.

Lemme	حَزَنَ	[ḥazana]
FF intermédiaire avec هُنَّ	حَزَنْنَ	
FF finale avec هُنَّ	حَزَنَّ	[hazanna]
FF : Forme Fléchie		

Figure 27 : Exemple de " حَزَنَ"

La présence de la lettre ت à la fin de la racine, engendre la fusion de celle-ci avec le suffixe ت de la 1ère personne singulier كَبَتُّ [kabattu] présenté dans la Figure 29.

Lemme	كَبَتَ	[kabata]
FF intermédiaire avec أنَا	كَبَتْتُ	
FF finale avec أنَا	كَبَتُّ	[kabattu]
FF : Forme Fléchie		

Figure 28 : Exemple de "كَبَتُّ "

Dans ces deux cas, l'opération « addAfter » prend l'argument "ّ " au lieu des suffixes normaux ن et ت.

VII. Critiques et propositions

Nous avons essayé d'appliquer les propositions de LMF à la langue arabe. Cependant, nous avons rencontré des difficultés aux niveaux des normes : ISO 12620 pour les catégories de données et ISO 24613 pour les ressources lexicales (LMF), sachant que la deuxième offre un méta modèle décoré par des catégories de données spécifiées dans la première.

Au niveau de la norme ISO 12620, ils ont proposé une sous-catégorisation qui doit être valable pour toutes les langues. Mais, malheureusement, elle n'est pas valable pour la langue arabe. Ce qui nous amène à proposer, au niveau conceptuel, de considérer la sous-catégorisation comme une information spécifique pour chaque langue et n'est pas comme une information générale.

Autrement dit, le lien récursif /broader/ de la classe Data Category nous permet de spécifier les catégories de données indépendamment de la langue. Néanmoins, la spécification peut être différente d'une langue à une autre, par exemple il y a une

proposition de sous catégorisation dans Eagles qui a défini trois catégories générales : nom, verbe et adjectif, mais pour la langue arabe, les adjectifs ce sont des sous catégories du nom.

Alors, nous proposons de remplacer le lien récursif [broader] par un autre lien entre la classe catégorie de données et la section de la langue comme le montre la Figure 15.

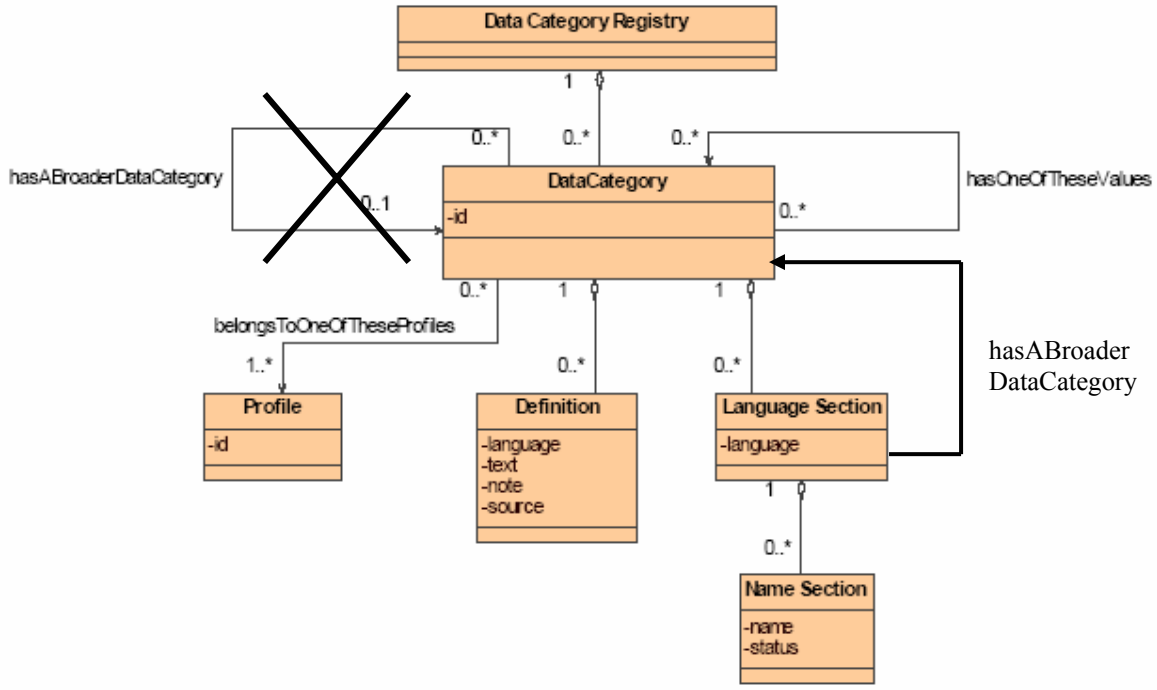


Figure 29 : Partie de la conception du RCD rectifié

Cette proposition de modification doit être discutée avec les responsables de l'ISO 12620, et actuellement, nous allons utiliser ce modèle rectifié pour représenter les catégories de données spécifiques à la langue arabe dans l'annexe A.

Au niveau de la norme ISO 24613, il n'y a pas une extension qui traite la phonologie qui joue un rôle très important dans la langue arabe. Devant ce manque, nous allons essayer de traiter les cas particuliers qui nécessitent l'intervention des règles phonologiques, par la multiplication des paradigmes. Néanmoins, cette solution ne reflète pas la réalité de la conjugaison des verbes arabes.

VIII. Conclusion

L'étude que nous avons réalisée dans ce chapitre a concerné l'application du noyau et des extensions morphologique et de mode de flexion de LMF sur la langue arabe. Nous avons fixé les catégories de données nécessaires pour chaque classe du modèle. En effet, au niveau du noyau, nous avons choisi les différents sous catégories du nom, les particules, les verbes et leurs formes dérivées comme entrée lexicale.

Nous remarquons qu'il y a des catégories de données liées au niveau syntaxique qui influencent la morphologie du mot arabe telle que la flexion casuelle. Par ailleurs, la sous-catégorisation proposée par la norme ISO12620 est différente de celle proposée par les grammairiens arabes.

Les paradigmes de flexion facilitent la génération des formes fléchies et réduisent la taille de la base de données. Cependant, le chevauchement entre les différents niveaux linguistiques, notamment morphologique et phonologique dans la langue arabe et l'absence de la phonologie dans la norme, montrent que ces paradigmes nécessitent une investigation assez détaillée.

CHAPITRE

4

Les paradigmes de flexion des verbes arabes selon LMF

I. Introduction

Comme nous avons mentionné dans le chapitre précédent, le modèle des paradigmes proposé dans LMF est simple et intéressant, néanmoins la conjugaison des verbes arabes nécessite parfois l'application des règles phonologiques qui ne sont pas représentées dans LMF.

Devant l'absence d'une extension phonologique dans LMF et des ressources linguistiques arabes qui réunissent toutes les règles phonologiques, nous allons travailler, sur les ressources existantes (les tableaux de El-Dahdah **[El-Dahdeh, 1999]** et de Ammar **[Ammar et al, 1999]**) qui contiennent un travail pertinent dans ce domaine d'investigation littéraire, afin d'appliquer les propositions de normalisation des modes de flexion à la langue arabe. Ces ressources traitent les influences des règles phonologiques sous forme des paradigmes. Elles ont valorisé les consonnes sensibles aux changements de leurs voyelles (précédente et suivante) et elles ont négligé ceux qui sont sensibles à l'ajout d'un suffixe.

Pour être plus méthodique, nous allons raffiner les classements des verbes arabes proposés dans les ressources existantes, en valorisant les consonnes qui peuvent être fusionnées avec les suffixes.

Nous allons commencer par un aperçu sur les principes de conjugaison, ensuite nous classerons les verbes selon la nature de la racine pour mettre en valeur les cas particuliers qui nécessitent l'intervention des règles phonologiques. Ces différentes classes de racine seront combinées avec les schèmes verbaux, pour aboutir enfin aux paradigmes de flexion tout en précisant les opérations à appliquer pour calculer les formes fléchies. Nous allons clôturer le chapitre par une section pour critiquer les résultats obtenus et proposer une solution qui représente la réalité de la langue arabe et peut enrichir les propositions de LMF.

II. Aperçu sur les principes de conjugaison

La conjugaison d'un verbe arabe est l'ensemble des formes que peut prendre ce verbe à la voix active et passive. Ces formes varient avec le mode et l'aspect (accompli, inaccompli indicatif, inaccompli subjonctif, inaccompli apocopé et l'impératif). Elles varient aussi avec la personne ou les personnes représentées par le sujet : كَتَبْتُ [katab**tu],** كَتَبْتُمَا **[**katb**tumâ]**…

Les flexions du verbe sont identiques dans les formes dérivées et dans le verbe nu. Le tableau suivant présente les marques de personnes, nombre et genre dans les trois aspects :

	1ère personne	2ème personne		3ème personne	
		masculin	Féminin	masculin	féminin
Singulier	--تُ	--تَ	--تِ	--َ	--َتْ
Duel	--نَا	--تُمَا		--َا	--َتَا
Pluriel		--تُمْ	--تُنَّ	--ُوا	--نَ

Tableau 10 : Flexion de l'accompli

	1ère personne	2ème personne		3ème personne	
		masculin	Féminin	masculin	féminin
Singulier	أ--ُ	تَ--ُ	تَ--ِينَ	يَ--ُ	تَ--ُ
Duel	نَ--ُ	تَ--َانِ		يَ--َانِ	تَ--َانِ
Pluriel		تَ--ُونَ	تَ--نَ	يَ--ُونَ	يَ--نَ

Tableau 11 : Flexion de l'inaccompli indicatif

	1ère personne	2ème personne		3ème personne	
		masculin	Féminin	masculin	féminin
Singulier	أ--َ	تَ--َ	تَ--ي	يَ--َ	تَ--َ
Duel	نَ--َ	تَ--َا		يَ--َا	تَ--َا
Pluriel		تَ--ُوا	تَ--نَ	يَ--ُوا	يَ--نَ

Tableau 12 : Flexion de l'inaccompli subjonctif

	1ère personne	2ème personne		3ème personne	
		masculin	Féminin	masculin	féminin
Singulier	أ--ْ	تَ--ْ	تَ--ي	يَ--ْ	تَ--ْ
Duel	نَ--ْ	تَ--َا		يَ--َا	تَ--َا
Pluriel		تَ--ُوا	تَ--نَ	يَ--ُوا	يَ--نَ

Tableau 13 : Flexion de l'inaccompli apocopé

Ces quatre tableaux (10, 11, 12, et 13) sont à la voix active. Cependant, les flexions de la voix passive sont différentes au niveau du préfixe dont la voyelle est *damma* ـُ au lieu de *fatha* ـَ tel que يَ--ُوا à l'actif sera يُ--ُوا au passif.

	2ème personne	
	masculin	Féminin
Singulier	اِ - ْ	اِ - ي
Duel	اِ - - َا	
Pluriel	اِ - ُوا	اِ - - نَ

Tableau 14 : Flexion de l'impératif

L'impératif ne se conjugue qu'à la 2ème personne. Il se rapproche de l'inaccompli apocopé et il n'est différent que par l'absence des préfixes. Dans ce cas, on peut trouver plusieurs verbes qui commencent par une consonne pourvue du symbole d'absence de voyelle *sukûn* ـْـ. En effet, il faut ajouter *hamza wasliya* (هَمْزَة وَصْلِيَّة) pour éviter la prononciation de deux consonnes successives (اِضْرِبْ [iḍrib]).

En générale, nous avons 19 schèmes verbaux, chacun d'eux a un paradigme, sauf deux schèmes « فَعَلَ » et « فَعِلَ » qui donnent plus qu'un paradigme (voir sectionV). Alors, nous remarquons que les paradigmes de conjugaison sont simples et leur nombre est réduit.

Pour certains verbes, c'est non seulement l'ajout d'un préfixe ou d'un suffixe, ou la transformation des voyelles qui permettent le calcul d'une forme fléchie, mais aussi des altérations importantes que peut subir un radical surtout pour les verbes défectueux, les verbes *hamzés*[5] et les verbes dupliqués. En effet, les changements du radical ne sont pas liés à la conjugaison, mais ils sont les conséquences d'application des règles phonologiques qui peuvent être apparues après les changements des voyelles ou l'ajout d'un suffixe.

Jusqu'à présent, il n'y a pas un travail qui réunit toutes les règles phonologiques, néanmoins il y a des travaux qui classent les verbes selon la nature des consonnes de la racine. Pour ce faire, ils ont considéré les consonnes qui nécessitent l'intervention des règles phonologiques après le changement des voyelles, comme un critère de distinction des racines. Par conséquent, chaque type de racine aura un nouveau paradigme. Mais, ils ont occulté les consonnes qui nécessitent l'intervention des règles phonologiques après l'ajout d'un suffixe.

En vue, d'appliquer le modèle de LMF, nous allons proposer, dans le paragraphe IV, un classement, pour les verbes arabes, selon la nature des consonnes de la racine, en considérant les changements des voyelles et l'ajout d'un suffixe pour combler l'absence de règles phonologiques dans le modèle de LMF.

[5] La racine du verbe comporte un hamza.

III. Critères de distinction des paradigmes

Un paradigme de flexion est une description détaillée des étapes de construction de toutes les formes fléchies d'un lemme donné. La distinction des paradigmes se fait selon la catégorie grammaticale (dans ce chapitre on s'intéresse à la catégorie du verbe seulement), le nombre des consonnes de la racine (selon longueur 3 ou 4) et le schème. En plus, l'absence des règles phonologies dans LMF, nous oblige de représenter les changements engendrés par ces règles par de nouveaux paradigmes. Nous allons classer les racines selon la nature de leurs consonnes et de leurs changements, afin d'avoir la nature de la racine comme un critère de distinction. Notons qu'il existe un autre critère d'ordre sémantique qui peut influencer l'application d'un paradigme. En effet, selon le sens du verbe, on applique le passif et l'actif (exemple نَسِيَ) ou on se limite à l'actif (exemple بَقِيَ). Nous avons résolu cette distinction à travers l'ajout de la catégorie de donnée /havePassive/ associée au lemme.

IV. Classification des racines

Nous allons classer les racines des verbes arabes selon la nature des consonnes puisqu'il y a des consonnes qui subissent des transformations ou des remplacements selon des règles phonologiques qui seront nécessaires soit après la modification des voyelles, soit après l'ajout d'un suffixe, (en vue d'alléger leur prononciation).

Nous distinguons plusieurs niveaux de classification, en se basant sur : la duplication des consonnes, la présence des consonnes défectueuses et la lettre hamza, la position de ces consonnes et leurs nombres dans la racine. En plus, nous avons limité les terminaisons qui peuvent être fusionnées avec les suffixes. Il s'agit des lettres ت et ن que nous les considérons comme un critère de distinction. La Figure 30, montre la nouvelle classification des verbes arabes.

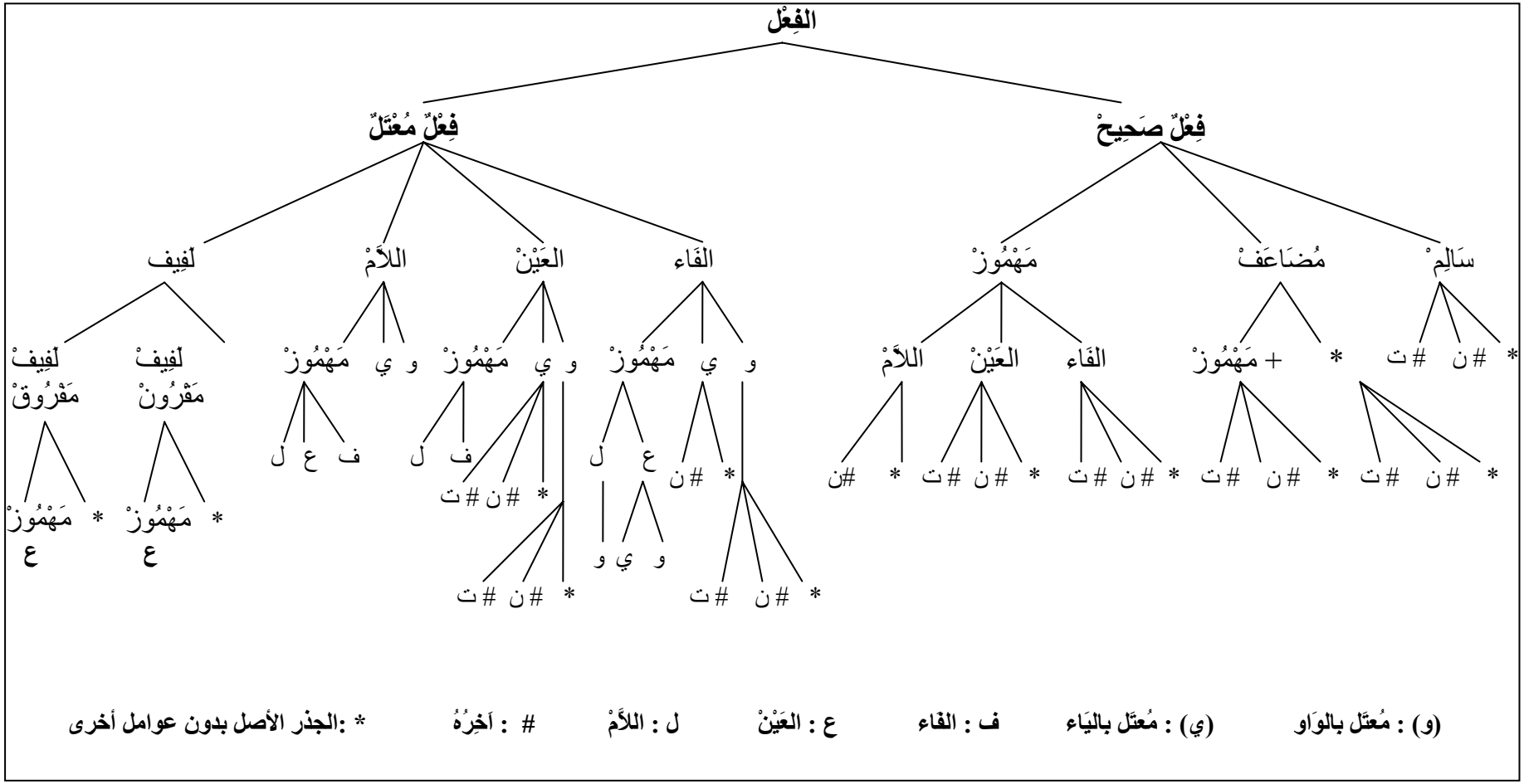


Figure 30 : Classification des verbes arabes selon la nature des consonnes de la racine

1. Les verbes sains (صَحِيحْ)

La racine d'un verbe sain ne prend ni و ni ي. Mais elle peut avoir *hamza* ou la duplication d'une consonne (le lemme du verbe aura le symbole *šadda*). Un *hamza* peut avoir des graphies différentes au cours de la conjugaison. Pour les formes fléchies ayant *hamza* dans la racine, soit elles gardent la graphie du *hamza* comme le lemme, soit elles le remplacent par une autre graphie dépendant de sa voyelle et de sa position (rarement il sera supprimé). Souvent, la graphie du *hamza* change au cours du calcul d'une forme fléchie. Pour cette raison, nous avons trois classes pour les verbes *hamzés* qui dépendent de la position du *hamza* dans la racine.

En plus, nous avons une classe qui réunit les deux cas particuliers *hamza* et *šadda,* une autre pour les verbes avec *šadda* et une classe qui n'a ni le *hamza* ni le *šadda*. Ce qui va engendrer l'apparition de six sous-classes.

1.1. Verbe sans *hamza* et sans *šadda* سَالِمْ

La racine de ce verbe n'a pas de *hamza* et ses consonnes ne sont pas dupliquées, il n'y a pas de transformation. C'est la classe la plus simple tels que كَتَبَ [kataba], ضَرَبَ [ḍaraba], قَتَلَ [qatala]. Mais la présence de la consonne ن ou ت à la fin du lemme peut engendrer la fusion de cette consonne avec les suffixes.

- Dernière consonne ن : sera fusionnée avec le suffixe du féminin pluriel ن, tel que حَزَنَ [ḥazana], حَزَنَّ [hazanna].
- Dernière consonne ت : sera fusionnée avec le suffixe de la première personne du singulier تُ tel que ثَبَتَ [tabata], ثَبَتُّ [tabattu].

1.2. Verbe avec *šadda* (مُضَاعَفْ)

La racine de ce verbe a deux consonnes identiques successives qui seront représentées par une seule consonne surmontée par *šadda* dans le lemme tel que مَدَّ [madda].

a) Verbe avec *šadda* et sans *hamza*

Dans le calcul des formes fléchies de cette classe, soit on garde le lemme avec *šadda*, soit on le libère et on duplique la consonne précédente.

- Dernière consonne ن : sera fusionnée avec le suffixe du féminin pluriel ن après la suppression de *šadda* et la duplication de la consonne précédente ن, tels que رَنَّ [ranna], يَرْنِنَّ [yarninna].

▪ Dernière consonne ت : sera fusionnée avec le suffixe de la première personne du singulier تُ après la suppression de *šadda* et la duplication de la consonne précédente ت, tel que بَتَّ [batta], بَتَتُّ [batattu].

b) Verbe avec *šadda* et *hamza*

Un verbe de cette classe comporte les caractéristiques de la classe précédente et un *hamza* à la première position. Par conséquent, il y a un traitement pour le signe *šadda* et un autre pour la consonne *hamza*.

▪ Dernière consonne ن : c'est comme la classe précédente avec une transformation possible de la graphie de *hamza*, tel que أَنَّ ['anna], إِئْنِنَّ [i'ninna].

▪ Dernière consonne ت : c'est comme la classe précédente avec une transformation possible de la graphie de *hamza*, tel que أَتَّ ['atta], أَئَتُ ['atattu], أَؤُتُ ['a'uttu].

1.3. Verbe avec *hamza* : مَهْمُوزْ

Les verbes de cette classe ont un *hamza* dans leur racine. La graphie de cette consonne dépend de sa position, de sa voyelle et de la voyelle précédente. Sachant que elle peut être au début ou au milieu ou à la fin.

a) Première consonne de la racine *hamza* : مَهْمُوزْ الفَاء

En général, ce verbe garde la même graphie du *hamza* même si la voyellation de celui-ci change. Le *hamza* n'aura d'influence que lorsqu'il y a ajout d'un préfixe.

▪ Dernière consonne ن : sera fusionnée avec le suffixe du féminin pluriel ن, tel que أَدِنَ ['adina], أَدَنَّ ['adanna].

▪ Dernière consonne ت : sera fusionnée avec le suffixe de la première personne du singulier تُ, comme dans أَبَتَ ['abata], أَبَتُّ ['abattu].

b) Deuxième consonne de la racine *hamza* : مَهْمُوزْ العَيْنْ

La graphie du *hamza* change quand il y a modification de sa voyelle tel que سَأَلَ [sa'ala] *hamza* sur *alif* et سُئِلَ [su'ila] la graphie de *hamza* est changée car sa voyelle *fatha* (ـَـ) est remplacée par *kasra* (ـِـ).

▪ Dernière consonne ن : sera fusionnée avec le suffixe du féminin pluriel ن, tels que مَأَنَ [ma'ana], مُئِنَّ [mu'inna].

▪ Dernière consonne ت : sera fusionnée avec le suffixe de la première personne du singulier تُ, نَأَتَ [na'ata], نَأَتُّ [na'attu] à titre d'exemple.

c) Troisième ou quatrième consonne de la racine *hamza* : مَهْمُوزْ اللاَّمْ

Si ce verbe est trilitère alors le *hamza* est la dernière consonne de la racine tel que هَنَأَ [hana'a], et s'il est quadrilitère alors le *hamza* est la dernière ou l'avant dernière consonne de la racine tels que كَرْفَأَ [karfa'a] et سَمْأَلَ [sam'ala]. Il s'agit d'une sous classe relative au cas de la dernière consonne ن spécifique aux verbes quadrilitères tel que اِطْمَأَنَّ [iṭma'anna], اِطْمَأَنَنَّ [iṭma'nanna].

2. Les verbes défectueux (مُعْتَلّ)

La racine d'un verbe défectueux risque d'être transformée au cours de la conjugaison, puisqu'elle est formée par une ou deux lettres défectueuses (و et ي). La présence de ces lettres dans une racine peut engendrer des altérations importantes dans le verbe.

Nous allons classer ces verbes selon le nombre et la position des lettres défectueuses. En résultat, nous distinguons cinq sous-classes.

2.1. Première consonne de la racine défectueuse : مُعْتَلّ الفاء

La 1ère consonne de la racine peut être و ou ي qui ont deux processus de conjugaison différents. En plus, les verbes de cette classe peuvent avoir *hamza* à la 2ème ou à la 3ème consonne de la racine, ou avoir comme dernière consonne de la racine les lettres ن ou ت . Alors, nous distinguons six causes de transformation:

- 1ère consonne de la racine و : un bon nombre de ces verbes gardent cette lettre sans transformation puisque elle est en première position, mais il y a quelques exceptions où cette consonne est supprimée tel que وَعَدَ [wa'ada], يَعِدُ [ya'idu].
- 1ère consonne de la racine ي : en général, ces verbes gardent cette lettre sans transformation puisqu'elle est en première position, par exemple يَتَمَ [yatama] est traité comme ضَرَبَ [ḍaraba].
- 2ème consonne de la racine *hamza* : la racine subit le changement de la graphie de *hamza* en cas de changement de sa voyelle tel que le verbe وَأَدَ [wa'ada], يَئِدُ [ya'idu].
- 3ème consonne de la racine *hamza* : le même effet que la sous-classe précédente, sauf que la position de la transformation de *hamza* est différente.
- Dernière consonne ن : sera fusionnée avec le suffixe du féminin pluriel ن, tel que يَمُنَ [yamuna], يَمُنَّ [yamunna].
- Dernière consonne ت : sera fusionnée avec le suffixe de la première personne du singulier تُ, tel que وَقَتَ [waqata], وَقَتُّ [waqattu].

2.2. Deuxième consonne de la racine défectueuse : مُعْتَلَ العَيْنْ

La 2ème consonne de la racine peut être و ou ي qui ont deux processus de conjugaison différents. En plus, les verbes de cette classe peuvent avoir *hamza* à la 1ère ou à la 3ème consonne de la racine, ou la dernière consonne de la racine ن ou ت . Alors, nous distinguons six causes de transformation :

- 2ème consonne de la racine و : il s'agit du cas où cette consonne se présente dans le lemme sous la forme de *alif* qui reprend sa forme initiale de la racine au cours de la conjugaison ou elle sera supprimée comme dans قَالَ [qâla], يَقُولُ [yaqûlu], قُلْ[qul].
- 2ème consonne de la racine ي : il s'agit du cas où cette consonne se présente dans le lemme sous la forme de *alif* qui reprend sa forme initiale de la racine ou elle sera supprimée au cours de la conjugaison tel que خَافَ [hâfa], خِيفَ [hifa], خُفْتُ [huftu].
- 1ère consonne de la racine *hamza* : la graphie de cette consonne de la racine varie selon sa voyellation tel que le verbe آلَ ['âla], يَؤُولُ [ya'ûlu].
- 3ème consonne de la racine *hamza* : la même chose que la sous-classe précédente, mais la position de transformation de *hamza* est différente.
- Dernière consonne ن : sera fusionnée avec le suffixe du féminin pluriel ن, tel que كَانَ [kâna], يَكُنَّ [yakunna].
- Dernière consonne ت : sera fusionnée avec le suffixe de la première personne du singulier تُ, tel que فَاتَ [fâta], فُتُّ [futtu].

2.3. Troisième consonne de la racine défectueuse : مُعْتَلَ اللاَّمْ

La 3ème consonne de la racine peut être و ou ي qui ont deux règles phonologiques différentes. En plus, les verbes de cette classe peuvent avoir *hamza* à la 1ère ou à la 2ème consonne de la racine. Alors, nous distinguons quatre causes de transformation :

- 3ème consonne de la racine و : cette consonne se présente dans le lemme sous la forme de *alif mamdouda* "ا" qui reprend sa forme initiale de la racine ou elle sera supprimée ou remplacée par ي au cours de la conjugaison tel que دَعَا [da'â], يَدْعُو [yad'ûd], اُدْعُ [ud'u], دُعِيَ[du'iya].
- 3ème consonne de la racine ي : cette consonne se présente dans le lemme sous la forme de *alif maqsoura* "ى" qui reprend sa forme initiale de la racine ou elle sera supprimée au cours de la conjugaison tel que رَمَى [ramâ], رُمِيَ [rumiya], اِرْمِ [irmi].
- 1ère consonne de la racine *hamza* : la graphie de cette consonne de la racine varie selon sa voyellation tel que le verbe أَبَا ['abâ], يُؤْبَى [ya'bû].

▪ 2ème ou 3ème consonne de la racine *hamza* : la même chose que la sous-classe précédente, mais la position de transformation de *hamza* est différente. Nous remarquons que *hamza* ne figure à la 3ème consonne de la racine qu'avec les verbes quadrilitères tel que رَهْيَأَ [rahya'a].

2.4. Deux consonnes de la racine défectueuses : لفِيفْ

Les verbes de cette classe ont deux consonnes défectueuses dans la racine qui peuvent être séparées ou liées.

d) Deux consonnes de la racine défectueuses séparées : لفِيفْ مَفْرُوقْ

La 1ére et la 3ème consonne de la racine peuvent être و ou ي qui engendre des transformations au cours de la conjugaison. En plus, les verbes de cette classe peuvent avoir *hamza* à la 1ère ou à la 2ème consonne de la racine. Alors, nous distinguons deux causes de transformation :

▪ 1ère consonne de la racine و et 3ème consonne de la racineي : la 1ère consonne de la racineو peut être supprimée et le 3ème consonne de la racine ي se présente dans le lemme sous la forme de *alif maqsoura* "ى" qui reprend sa forme initiale ou elle sera supprimée au cours de la conjugaison tel que وَقَى [waqâ], أَقِي ['aqî], أَقْ ['aqi].

▪ 2ème consonne de la racine *hamza* : la graphie de cette consonne varie selon sa voyellation tel que le verbe وَأَى [wa'â], يَئِي [ya'î].

e) Deux consonnes de la racine défectueuses successives : لفِيفْ مَقْرُونْ

Le 2ème et le 3ème consonne de la racine peuvent être و ou ي. Généralement, une seul consonne sera transformée au cours de la conjugaison et l'autre traitée comme une consonne normale. Mais la seule différence c'est quand les deux consonnes sont ي, il se peut qu'elles soient réunis par *šadda* tel que أَحْيَا ['aḥyâ], يُحَيَّان [yuḥiyyâni].

V. Classification des schèmes verbaux

Les schèmes verbaux sont en nombre de 19 **[El-Dahdah, 1999]** qui sont présentés dans le Tableau 15. Ils peuvent être nus ou augmentés. En plus, il y a ceux qui sont spécifiques pour les verbes trilitères et d'autres pour les quadrilitères. Nous signalons que, le premier schème فَعَلَ peut avoir trois méthodes différentes de conjugaison selon la voyelle de la 2ème consonne de la racine de l'inaccompli يَفْعُلُ, يَفْعِلُ, et يَفْعَلُ alors que le troisième schème فَعِلَ donne deux méthodes différentes de conjugaison. Ainsi, les classes de schèmes deviennent en nombre 22.

Numéro	Classe de schème	
	Schème	Voyelle de la 2ème consonne à l'inaccompli actif
1	فَعَلَ	ـُ
2	فَعَلَ	ـِ
3	فَعَلَ	ـَ
4	فَعُلَ	ـُ
5	فَعِلَ	ـَ
6	فَعِلَ	ـِ
7	فَعَّلَ	ـِ
8	فَاعَلَ	ـِ
9	أَفْعَلَ	ـِ
10	تَفَعَّلَ	ـَ
11	تَفَاعَلَ	ـَ
12	اِنْفَعَلَ	ـِ
13	اِفْتَعَلَ	ـَ
14	اِفْعَلَّ	ـَ
15	اِسْتَفْعَلَ	ـِ
16	اِفْعَوْعَلَ	ـَ
17	اِفْعَوَّلَ	ـَ
18	اِفْعَالَّ	ـَ
19	فَعْلَلَ	ـِ
20	تَفَعْلَلَ	ـَ
21	اِفْعَنْلَلَ	ـِ
22	اِفْعَلَلَّ	ـِ

Tableau 15 : les classes de schème

VI. Classification des paradigmes de flexion

Le modèle proposé par LMF se base sur une description détaillée de chaque modification du lemme. Cependant, l'absence de règles phonologiques dans LMF a été l'origine de la duplication de certains paradigmes ayant le même principe de conjugaison, pour prendre en considération des transformations particulières d'ordre phonologique tels que la transcription du *hamza* ou terminaison de la racine par les lettres ن ou ت . En plus, le nombre des lettres d'un lemme peut être un facteur de duplication de paradigmes selon la position de transformations.

Partant des constatations sus indiquées, nous avons déjà classé les verbes selon la nature et le nombre des transformations possibles sur le lemme au cours de la conjugaison, ce qui a engendré l'apparition de 42 classes de racines verbales. Ces différentes classes de racines peuvent être combinées avec les 22 classes de schème comme le montre la Figure31 ci-après.

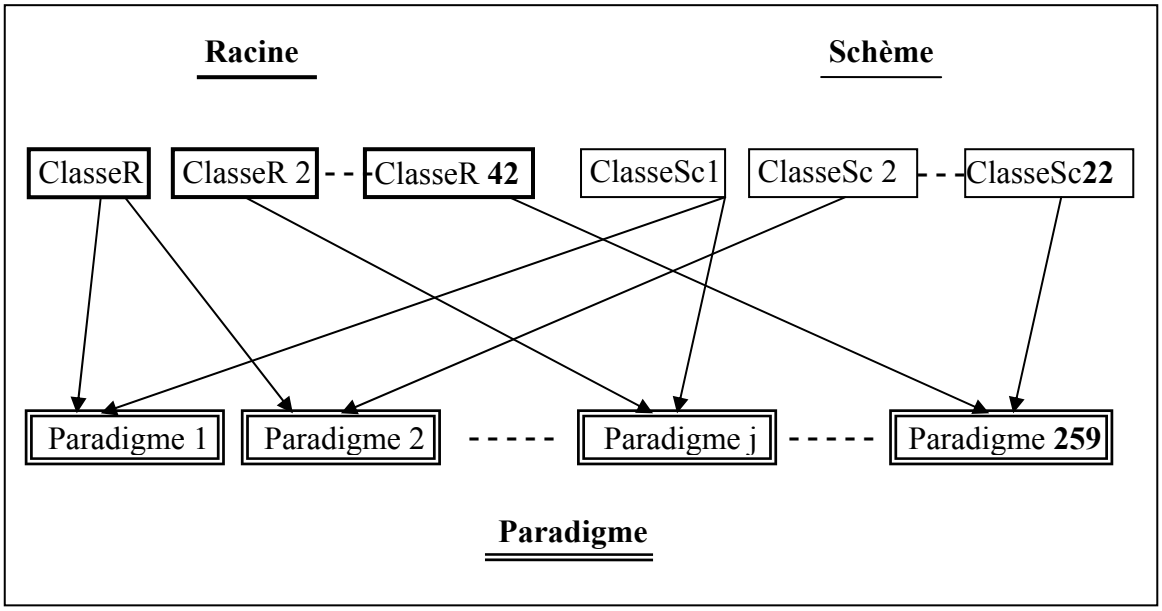


Figure 31 : Combinaison des classes de racine et des classes de schème

Nous allons présenter dans l'annexe B les différentes combinaisons possibles entre les classes de schème et les classes de racine en utilisant le troisième niveau de classement proposé dans la Figure 30. Dans l'intersection, nous mettrons les références des paradigmes convenables. Ces références seront réunies dans l'annexe C pour proposer un verbe type pour chaque référence.

La combinaison de 42 classes de racine avec 22 classes de schème donne un grand nombre de paradigme (924), mais en réalité, nous avons besoin seulement de 259 paradigmes comme le montre la Figure 31, ce nombre réduit s'explique par les raisons suivantes :

- Au niveau sémantique, il arrive de trouver, quoique de façon rare, des schèmes qui revêtent un sens assez particulier comme dans le cas de اِفْعَلَّ, alors que dans فَعَّلَ, il est très utilisé : اِفْعَلَّ n'est généré qu'à une seule classe de racine (**السَالِمْ**).
- Si deux schèmes ont le même nombre de lettres et les mêmes voyelles respectant les mêmes positions, nous leur créons un seul paradigme. c'est le cas de اِفْتَعَلَ et اِنْفَعَلَ.
- Les verbes ayant deux radicaux défectueux successifs sont traités comme les verbes qui ont une seule lettre défectueuse : ils ont le même paradigme.
- Les verbes quadrilitères qui ont des schèmes spécifiques sont rarement augmentés, d'autant plus que le nombre de classe des racines quadrilitères est très réduit : par exemple, tous les verbes quadrilitères ne peuvent jamais avoir le signe *šadda.*

Néanmoins, il est à noter que, malgré l'impossibilité de combinaison de quelques classes de schème avec quelques classes de racine et la possibilité de rencontrer un seul

paradigme d'une classe de racine pour deux schèmes différents, on trouve des combinaisons d'une classe de racine et d'une classe de schème qui ont deux paradigmes différents, à titre d'exemple :

- Le paradigme 241فَأى et le paradigme 242 رَأى (voir annexe C) : la différence entre ces deux paradigmes est due à la prononciation qui nécessite la suppression de أ dans le paradigme 242 : رَأى [ra'â], يَرَى [yarâ].
- La racine ط ي ر [ṭ y r] avec le schème تَفَعَّلَ donne deux lemmes : تَطَيَّرَ qui est le résultat de mécanisme de dérivation et un autre lemmeاِطَّيَّرَ qui a subi la fusion de ت et ط et l'ajout de *hamza wasliya* اِ pour éviter la présence de *sukûn* au début du mot.

VII. Application des opérations

Nous allons spécifier les opérations nécessaires pour les principales classes des verbes (le deuxième niveau de classement proposé dans la Figure 30). Nous allons utiliser les opérations proposées dans LMF : *add, addAfter, addBefore, removeAfter, remove, substitute* et *copy*. Ces opérations vont être utilisées par ordre pour le calcul des formes fléchies à partir du lemme.

1. Cas des verbes sains (صَحِيحْ)

Nous rappelons que cette classe a six sous-classes que nous allons spécifier leurs opérations. Nous signalons qu'il y a des classes qui ont les mêmes opérations et les mêmes arguments mais qui différent selon la position.

1.1. Les opérations du verbe sans *hamza* et sans *šadda* فِعْلَ سَالِمْ

Cette sous-classe est la plus simple. Elle englobe le plus grand nombre de verbe, elle a plusieurs racines qui se terminent par ن ou ت et elle est combinée à toutes les classes de schème. En total, nous avons 58 paradigmes qui utilisent les opérations suivantes :

« substitute » : pour remplacer les voyelles,

« addBefore » : pour ajouter les préfixes à l'inaccompli,

« addAfter » : pour ajouter les suffixes.

« removeAfter » : pour supprimer la voyelle finale.

Dans la Figure 32, nous avons utilisé ces opérations pour le calcul d'une forme fléchie.

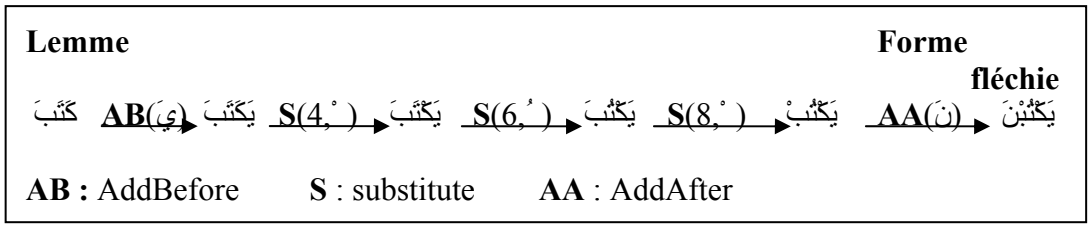


**Figure 32 : Calcul de la forme fléchie "يَكْتُبْنَ" **

Par conséquent, la graphie des consonnes de la racine ne sera jamais modifiée. En plus, si la dernière consonne de la racine est ت, on utilise « addAfter » pour ajouter à la fin *šadda* au lieu du suffixe de la première personne du singulier ت comme dans la Figure 33.

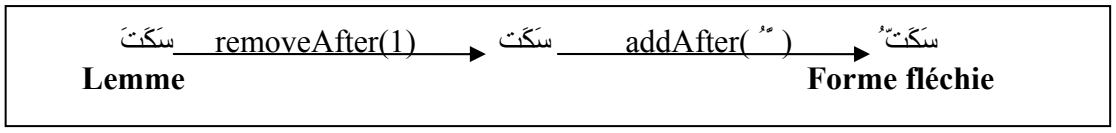


Figure 33 : Calcul de la forme fléchie " سَكَتَ "

Idem pour la dernière consonne de la racine ن, où on utilise « addAfter » pour ajouter à la fin *šadda* au lieu du suffixe du féminin pluriel ن comme dans l'exemple de la Figure 34.

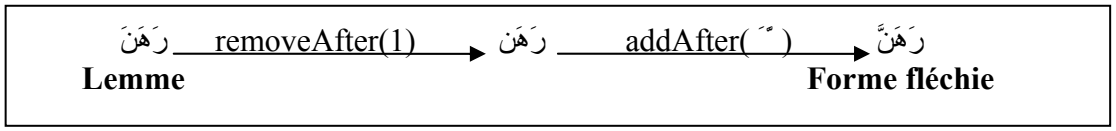


Figure 34 : Calcul de la forme fléchie "رَهَنَّ"

1.2. Les opérations du verbe avec *šadda* (مُضَاعَفْ)

a) Les opérations du verbe avec *šadda* et sans *hamza* (مُضَاعَفْ)

Cette sous-classe a le signe *šadda* qui peut avoir un traitement particulier : en cas de suppression, on doit dupliquer la consonne précédente et ajouter une voyelle à la première consonne. Elle est spécifique pour les verbes trilitères, généralement le *šadda* remplace les deux derniers radicaux. Il aura encore un traitement particulier si la consonne surmontée est ن ou ت. En résultat, nous avons 27 paradigmes (numéroté de 59 à 85) qui utilisent les opérations suivantes :

« substitute » : pour remplacer les voyelles,

« add Before » : pour ajouter les préfixes à l'inaccompli,

« add After » : pour ajouter les suffixes,

« remove » : pour supprimer le *šadda,*

« removeAfter » : pour supprimer la voyelle finale.

« copy » : pour dupliquer la consonne qui a été surmontée par *šadda,*

« add » : pour ajouter une voyelle à la consonne en question.

Dans la Figure 35, nous avons utilisé ces opérations pour le calcul des deux formes fléchies ayant les mêmes traits morphologiques.

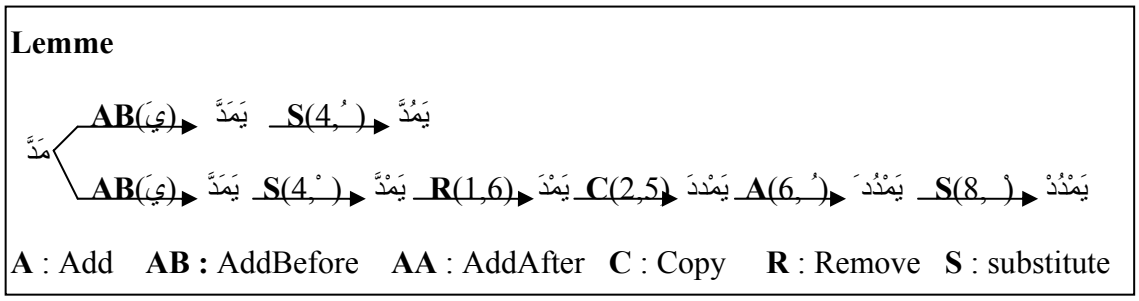


Figure 35 : Calcul des formes fléchies : "يَمُدَّ " et "يَمْدُدْ "

En plus, si la consonne surmontée par *šadda* est ت comme dans la Figure 36, on utilise d'abord les opérations précédentes, ensuite, on ajoute à la fin *šadda* par l'opération « addAfter » au lieu du suffixe de la première personne de singulier ت.

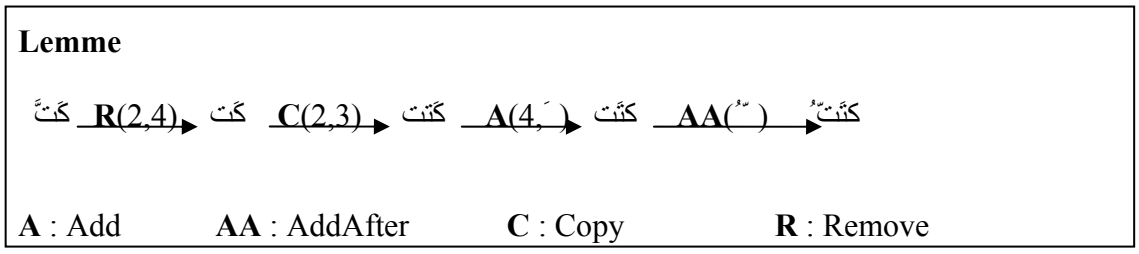


Figure 36 : Calcul de la forme fléchie : " كتَتّ "

Si la consonne surmontée par *šadda* est ن comme dans la Figure 37, on utilise les opération précédentes, ensuite, on ajoute à la fin *šadda* par l'opération « add after » au lieu du suffixe du féminin pluriel ن.

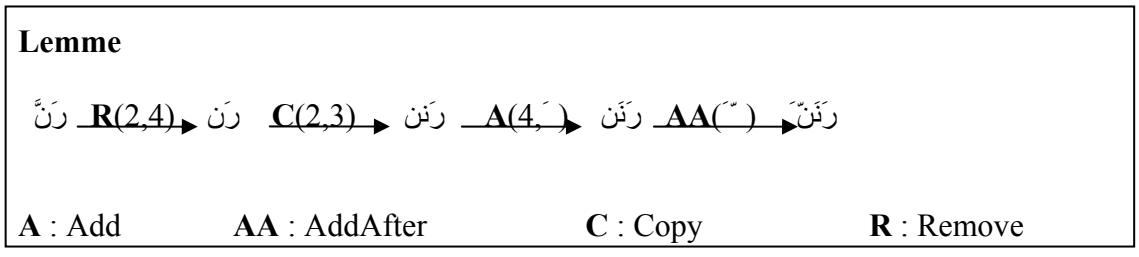


Figure 37 : Calcul de la forme fléchie : " رَنَنّ "

b) Les opérations du verbe avec *šadda* et *hamza*

Cette sous-classe comporte les caractéristiques des deux sous-classes : verbe avec *šadda* et verbe avec *hamza*. Alors, il y a un traitement pour le signe *šadda* déjà présenté dans la classe précédente et un autre pour la consonne *hamza*. En résultat, nous avons 12 paradigmes.

Généralement, *hamza* est en première position. Il sera remplacé par une autre graphie s'il a changé de voyellation et si le verbe a un préfixe. Dans ce cas, nous utilisons l'opération

« substitute » pour le remplacement comme dans l'exemple de la Figure 38 qui présente le calcul de deux formes fléchies ayant les mêmes traits morphologiques.

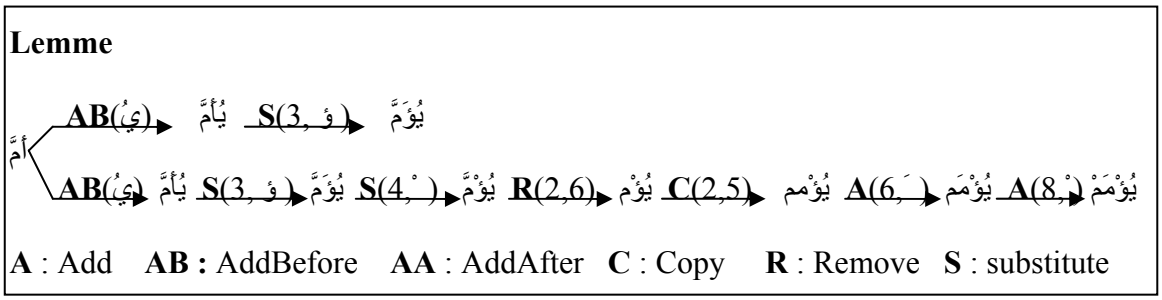


Figure 38 : Calcul des formes fléchies : " يُؤَمَّ " et " يُؤْمَمْ "

En plus, dans cette sous-classe, il y a des verbes qui se terminent par ن ou ت. Cette terminaison sera traitée comme la terminaison des verbes avec *šadda*.

1.3. Les opérations du verbe avec *hamza*

a) Les opérations du verbe ayant la 1ère consonne de la racine *hamza*

Cette sous-classe a un seul traitement particulier lié à la présence de *hamza* au début qui change de graphie s'il y a un préfixe et s'il subit un changement de voyellation. Généralement, nous appliquons les mêmes principes que pour la sous-classe des verbes sans *hamza* et sans *šadda*, en plus, nous utilisons l'opération « substitute » pour remplacer la graphie de *hamza* par une autre, ou « remove » pour supprimer *hamza* dans le but d'alléger la prononciation. En résultat, nous avons crée 23 paradigmes.

Dans la Figure 39, nous présentons le calcul de deux formes fléchies n'ayant pas les mêmes traits morphologiques. La première forme fléchie est à l'inaccompli passif pour montrer la modification de la graphie de *hamza* et la deuxième est à l'impératif pour montrer la suppression de celle-ci.

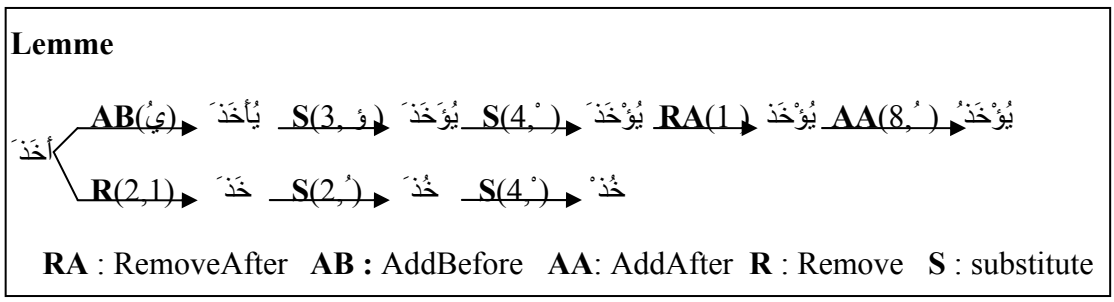


**Figure 39 : Calcul des formes fléchies : " يُؤْخَذ" et "خُذْ " **

En plus, dans cette sous-classe, il y a des verbes qui se terminent par ن ou ت. Cette terminaison sera traitée comme la terminaison des verbes sans *hamza* et sans *šadda* (فِعْلَ سَالِمْ).

b) Les opérations du verbe ayant la 2ème consonne de la racine *hamza* :

Cette sous-classe a un seul traitement particulier lié à la présence de *hamza* au milieu. Il change de graphie s'il subit un changement de voyellation. Généralement, nous appliquons les mêmes principes que pour la sous-classe des verbes sans *hamza* et sans *šadda*. En plus, nous utilisons l'opération « substitute » pour remplacer la graphie de *hamza* par une autre, ou « remove » pour supprimer *hamza* dans le but d'alléger la prononciation. En résultat, nous avons crée 18 paradigmes.

Dans la Figure 40, nous présentons le calcul deux formes fléchies n'ayant pas les mêmes traits morphologique comme dans l'exemple de la sous-classe précédente, mais la seule différence c'est au niveau de la position de l'application des opérations.

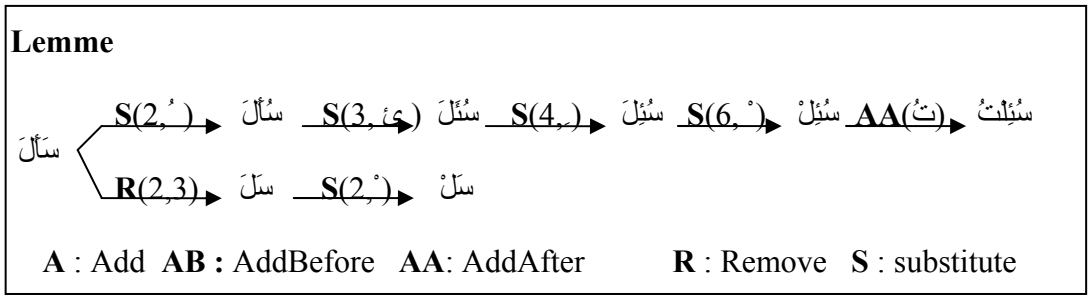


Figure 40 : Calcul des formes fléchies : " سُئِلْتُ " et " سَلْ "

En plus, dans cette sous-classe, il y a des verbes qui se terminent par ن ou ت. Cette terminaison sera traitée comme la terminaison des verbes sans *hamza* et sans *šadda* (فِعْلَ سَالِمْ).

c) Les opérations du verbe ayant la 3ème ou la 4ème consonne de la racine *hamza*

Cette sous-classe a un seul traitement particulier lié à la présence de *hamza* à la fin qui change de graphie s'il subit un changement de voyellation. Généralement, nous appliquons les mêmes principes que pour la sous-classe des verbes sans *hamza* et sans *šadda*. En plus, nous utilisons l'opération « substitute » pour remplacer la graphie de *hamza* par une autre. En résultat, nous avons crée 19 paradigmes.

Dans la Figure 41, nous présentons le calcul de deux formes fléchies n'ayant pas le même lemme pour des verbes quadrilitères. Le premier lemme a un *hamza* à la 3ème position et le deuxième a un *hamza* à la 4ème position.

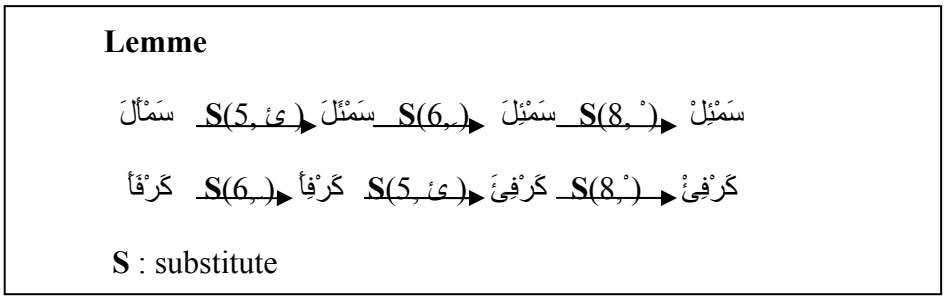


Figure 41 : Calcul des formes fléchies : "سَمْئِلْ " et " كَرْفِئْ"

En plus, dans cette sous-classe, il y a des verbes qui se terminent par ن . Cette terminaison sera traitée comme la terminaison des verbes sans *hamza* et sans *šadda* (فِعْلَ سَالِمْ).

2. Cas des verbes défectueux (المُعْتَلْ)

Nous rappelons que cette classe a cinq sous-classes. Nous allons spécifier les opérations de chaque sous-classe et nous signalons qu'il y a des classes qui ont les mêmes opérations qui différent selon la position.

2.1. Les opérations du verbe ayant la 1ère consonne de la racine défectueuse

Cette sous-classe se caractérise par la présence d'une lettre défectueuse à la première position. Généralement, cette lettre est traitée comme une consonne normale, mais quelques cas particuliers subsistent lors de la suppression ou de la modification de cette lettre. Nous utilisons alors, en plus des opérations de la sous-classe des verbes sans *hamza* et sans *šadda* (فِعْلَ سَالِمْ), l'opération « remove » pour supprimer cette lettre ou l'opération « substitute » pour la remplacer. Dans la Figure 42, nous présentons le calcul de deux formes fléchies pour des verbes n'ayant pas le même lemme. Le premier lemme a la lettre défectueuse و et le deuxième a la lettre défectueuse ي.

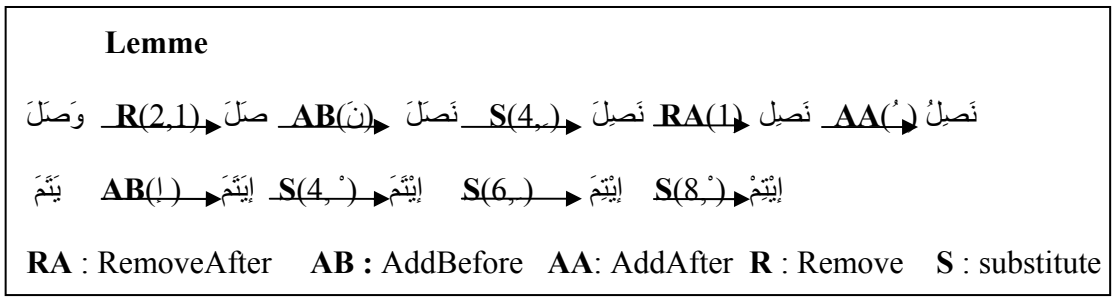


Figure 42 : Calcul des formes fléchies : " نَصِلُ" et " إيْتِمْ"

Cette sous-classe peut avoir encore *hamza* en 2ème ou 3ème consonne de la racine ou une des deux consonnes ن et ت à la fin du verbe. Ces quatre cas nécessitent un traitement particulier avec le traitement de la lettre défectueuse. Ainsi, nous avons 23 paradigmes avec la réutilisation des paradigmes précédents en cas de stabilité de la racine. Dans la

Figure 43, nous présentons un exemple de ces cas particuliers qui a un *hamza* et une lettre défectueuse.

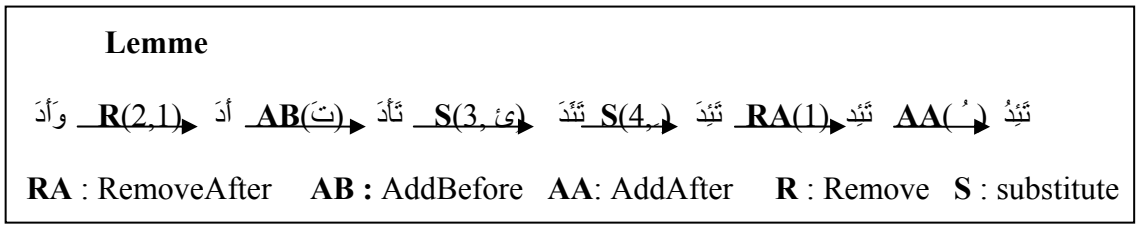


Figure 43 : Calcul de la forme fléchie " تَئِدُ "

2.2. Les opérations du verbe ayant la 2ème consonne de la racine défectueuse

Cette sous-classe se caractérise par la présence d'une lettre défectueuse à l'intérieur du verbe. Généralement, cette lettre est transformée en *alif mamdouda* "ا" dans le lemme et au cours de la conjugaison elle reprend sa forme initiale ou bien elle sera supprimée. Alors, nous utilisons, en plus des opérations de la classe des verbes sains, l'opération « remove » pour supprimer cette lettre ou l'opération « substitute » pour la remplacer. Cette classe peut avoir encore *hamza* comme 1ère ou 3ème consonne de la racine ou une des deux consonnes ن et ت à la fin de la racine. Ces quatre cas nécessitent un traitement particulier avec le traitement de la lettre défectueuse. En résultat, nous avons 32 paradigmes avec la réutilisation des paradigmes précédents en cas de stabilité de la racine (racine inchangée).

Dans la Figure 44, nous présentons le calcul deux formes fléchies pour des verbes n'ayant pas le même lemme mais ayant les mêmes traits morphologiques. Le premier lemme a la lettre défectueuse و et le deuxième a la lettre défectueuse ي.

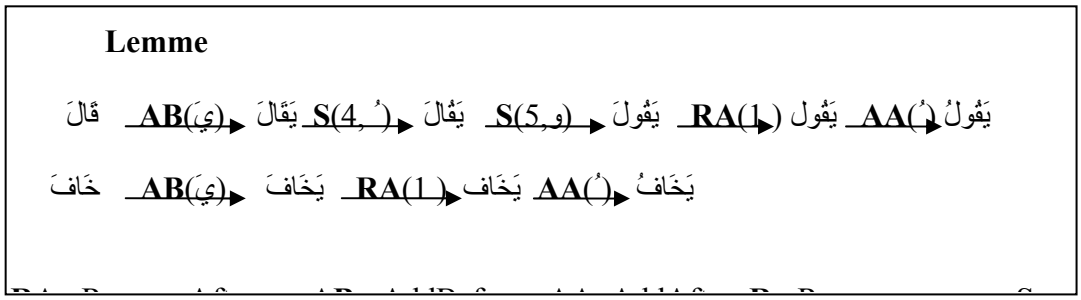


Figure 44 : Calcul des formes fléchies : " يَقُولُ " et " يَخَافُ "

2.3. Les opérations du verbe ayant la 3ème consonne de la racine défectueuse

Cette sous-classe se caractérise par la présence d'une lettre défectueuse à la fin du verbe. Généralement, cette lettre est transformée en *alif maqsoura* "ى" dans le lemme alors qu'au cours de la conjugaison elle reprend sa forme initiale ou elle sera supprimée. Alors, nous utilisons, en plus des opérations de la classe des verbes sains, l'opération « remove » pour

supprimer cette lettre ou l'opération « substitute » pour la remplacer. Cette classe peut avoir encore *hamza* comme 1ère ou 2ème consonne de la racine. Ces deux cas nécessitent un traitement particulier avec celui de la lettre défectueuse. En résultat, nous avons 36 paradigmes qui dépendent de la nature de la lettre défectueuse.

Dans la Figure 45, nous présentons le calcul de deux formes fléchies pour des verbes qui n'ont pas le même lemme mais ayant les mêmes traits morphologiques. Le premier lemme a la lettre défectueuse و et le deuxième a la lettre défectueuse ي.

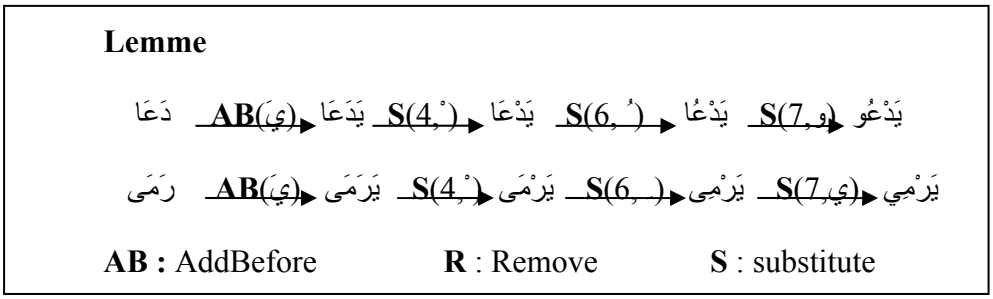


Figure 45 : Calcul deux formes fléchies : "يَدْعُو " et "يَرْمِي "

2.4. Les opérations du verbe ayant deux consonnes défectueuses

Normalement, les verbes de cette classe sont contrôlés par des règles phonologiques qui peuvent être utilisées pour les deux consonnes défectueuses, mais par fois, il y a pas recours à ces règles car leurs conditions ne sont pas satisfaites surtout lorsque une de ces consonnes est à la première position.

a) Les opérations du verbe ayant deux consonnes de la racine défectueuses séparées

Cette sous-classe se caractérise par la présence de deux lettres défectueuses, une au début et une autre à la fin du verbe. Généralement, la 1ere lettre reste stable et la 2ème lettre est transformée en *alif mamqsûra* "ى" dans le lemme. Au cours de la conjugaison elle est traitée comme la classe précédente. Mais, il y a quelques exceptions où les deux lettres seront transformées. Cette classe peut avoir encore *hamza* en 2ème consonne de la racine. Ce dernier cas nécessite un traitement particulier avec celui des deux lettres défectueuses. En résultat, nous avons 7 paradigmes avec la réutilisation des paradigmes précédents en cas de stabilité d'une des deux lettres défectueuses.

Dans la Figure 46, nous présentons le calcul de deux formes fléchies pour des verbes qui n'ont pas le même lemme mais ayant les mêmes traits morphologiques. Le premier lemme a deux lettres défectueuses séparées et le deuxième a deux lettres défectueuses et un *hamza*.

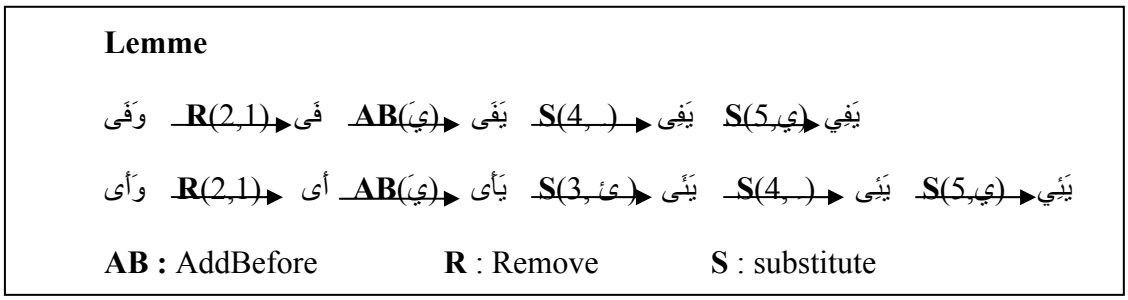


Figure 46 : Calcul des formes fléchies "يَفِي " et "يَئِي "

b) Les opérations du verbe ayant deux consonnes de la racine défectueuses successives

Cette classe se caractérise par la présence de deux lettres défectueuses à la fin du verbe. Généralement, la 1ere lettre reste stable et la 2ème lettre est transformée en *alif mamqsûra* "ى" dans le lemme. Au cours de la conjugaison, elle est traitée comme les verbes qui ont une seule consonne de la racine défectueuse à la 3ème position (مُعْتَلَ اللاَّمْ). Mais, il y a quelques cas qui ne sont pas traités comme la duplication d'une lettre défectueuse. Cette sous-classe peut avoir encore *hamza* en 1ère consonne de la racine, ce cas nécessite un traitement particulier avec le traitement d'une des deux lettres défectueuses. En résultat, nous avons seulement 4 paradigmes avec la réutilisation des paradigmes précédents dans la plupart des cas.

Dans la Figure 47, nous présentons le calcul de la forme fléchie d'un verbes ayant deux lettres défectueuses successives dans la racine.

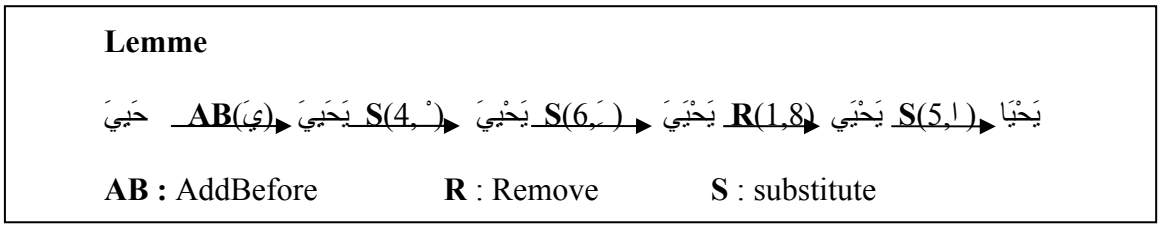


Figure 47 : Calcul de la forme fléchie "يَحْيَا "

VIII. Critiques et propositions

Dans la langue arabe, il y a un chevauchement entre les différents niveaux linguistiques. Nous avons remarqué ce phénomène en particulier dans les niveaux : morphologique et phonologique. Lorsqu'on applique le principe de conjugaison, il y a des règles phonologiques qui seront déclenchées après la modification d'une voyelle ou l'ajout d'un suffixe. En effet, nous avons essayé dans ce chapitre d'appliquer le modèle proposé par LMF. Nous avons généré 259 paradigmes qui est nombre très important, difficile à gérer et surtout ne reflète pas la réalité de la langue arabe.

Pour résoudre ces problèmes, nous proposons de considérer une nouvelle extension pour la phonologie dans LMF qui aura le modèle suivant présenté dans la Figure 48 :

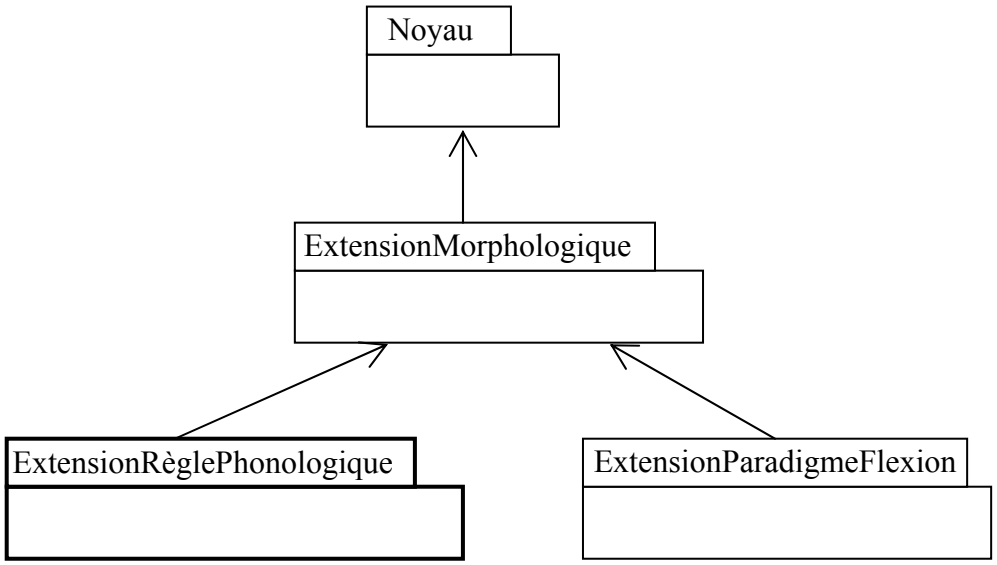


Figure 48 : Besoin d'une extenxion phonologique

A titre d'exemple, nous pouvons considérer dans cette extension phonologique les aspects d'une condition d'une règle phonologique qui concerne une ou de deux consonnes et leurs opérations nécessaires pour la transformation de la forme fléchie en question, qui seront organisées dans la modélisation de la Figure 49.

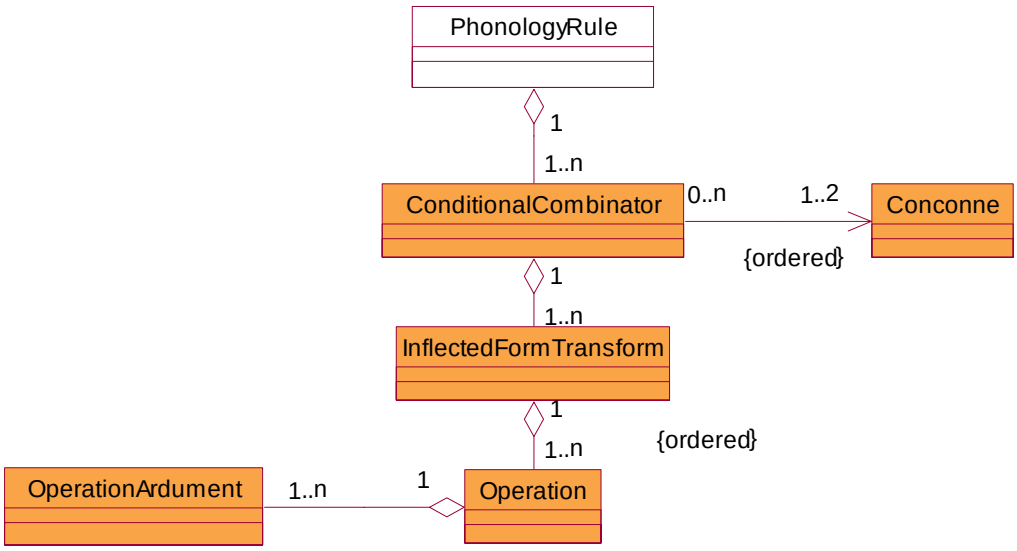


Figure 49 : Proposition d'un modèle pour l'extension phonologique

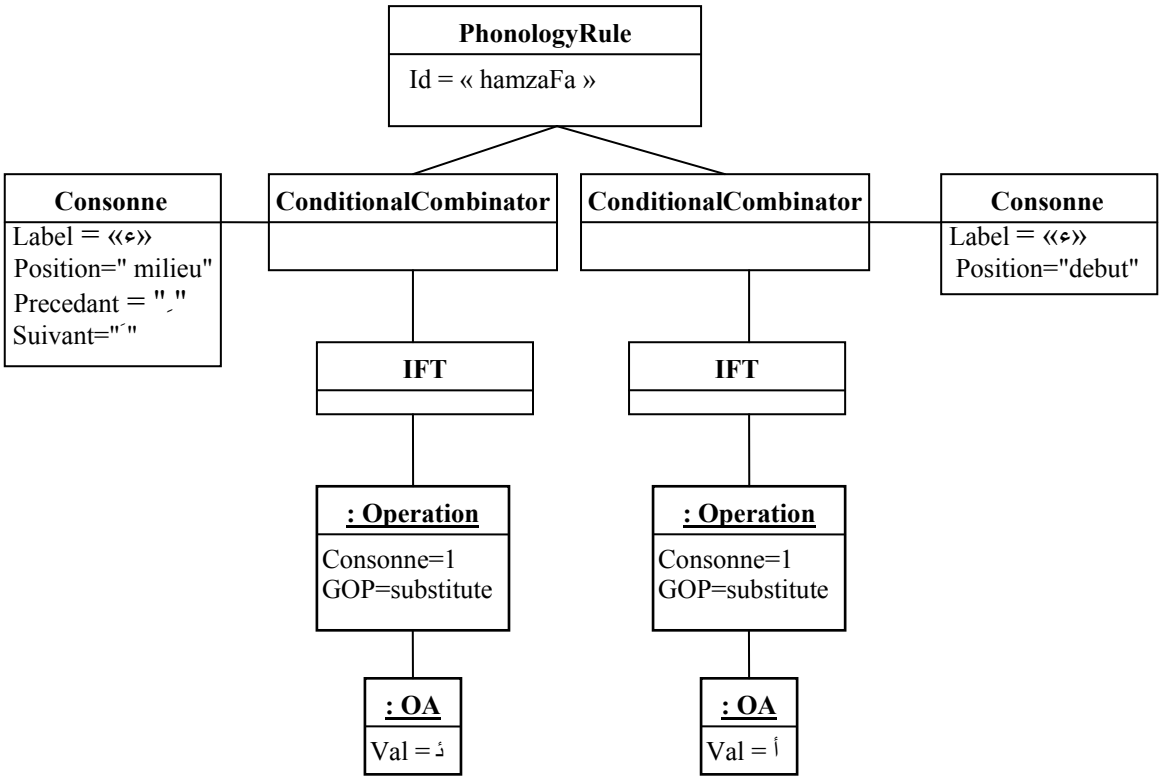
Dans ce modèle, nous présentons l'extension phonologique qui sera liée au noyau et aux autres extension à travers la classe *LemmatisedForm* qui peut avoir zéro ou un seul paradigme de flexion et zéro ou plusieurs règles phonologiques puisqu'il y a des verbes

qui n'ont pas besoin de règle phonologique tel que les verbes sains sans *hamza* et d'autres verbes qui ont besoin de plusieurs règles phonologiques tel que les verbes ayant à la fois *hamza* et la lettre défectueuse. Ainsi, la présence d'une règle phonologique, nous oblige de vérifier chaque nouvelle forme fléchie générée par les paradigmes de flexion, est ce qu'elle vérifie les conditions de cette règle ? Si ces conditions sont vérifiées, il faut appliquer les transformations liées à cette règle.

Dans la Figure 50, nous avons trois nouvelles classes :

- *ConditionalCombinator* : permet la combinaison entre un ou deux consonnes qui forment les conditions de la règle en question et les transformations qui seront appliquées à la forme fléchie. Par conséquent, elle fait référence à un ou deux *Consonne* et elle se compose par un ou plusieurs *InflectedFormTransform*.
- *Consonne* : il s'agit d'une consonne qui peut avoir plusieurs caractéristiques qui offre les conditions d'application d'une règle phonologique sur une forme fléchie. Ses caractéristiques sont classées parmi les catégories de données traitées par la norme ISO 12620, tels que le label et la position.
- *InflectedFormTransform* : regroupe un ensemble d'opération avec un ordre précis. Chaque opération n'est utilisée qu'une seule fois. Elle appartient à une seule *ConditionalCombinator.*

Nous signalons que les classes *Operation* et *OperationArgument* sont les mêmes classes de l'extension de mode de flexion. En plus, la cardinalité entre les classes *ConditionalCombinator* et *InflectedFormTransform*, nous permet d'avoir deux types de transformation différents sur la même forme fléchie, ce qui va nous donner deux formes différentes.



IFT=InflectedFormTransform
OA : OperationArgument **GOP** : GraphicalOperation

Figure 50 : Exemple d'une règle phonologique

Nous présentons dans la Figure 50 un exemple d'une règle phonologique qui réunit les conditions et les opérations de transformation de la forme fléchie qui seront appliquées à la position de la consonne en question.

IX. Conclusion

Nous avons proposé, dans ce chapitre, un classement des verbes selon les racines et les schèmes qui nous a aidé à extraire les paradigmes nécessaires pour plus que 16000 verbes. Ces paradigmes peuvent être appliqués totalement ou partiellement sur un verbe en fonction d'une catégorie de données qui spécifie s'il admet seulement la voix active ou s'il admet les deux voix (active et passive).

Selon LMF, nous avons fixé les opérations qui permettent de calculer les différentes formes fléchies associées à chaque paradigme. En résultat, nous avons 259 paradigmes qui est un nombre très important et ne présente pas la réalité de la langue arabe, ce qui nous a poussé de proposer une nouvelle extension qui traite la phonologie. Cette proposition va présenter la réalité de la langue arabe et le nombre de paradigme sera réduit à une vingtaine.

CHAPITRE

5

Conception de la base ArabicLDB

I. Introduction

Après avoir étudié l'applicabilité de LMF à la langue arabe, nous présentons dans ce chapitre la conception de base *ArabicLDB* (Arabic Lexical DataBase) qui est la réalisation de cette application, représentant l'extension morphologique et celle des paradigmes de flexion selon LMF. Cette conception sera un raffinement du méta-modèle proposé par LMF, par la spécification des principales catégories de données de la langue arabe, en vue de préciser la structure de cette base et mettre en valeur nos choix linguistiques. En plus, nous avons besoin d'un gestionnaire qui nous permet de créer et d'exploiter cette base, afin de tester son contenu. La conception de la base et de son gestionnaire est réalisée par le langage standard UML qui comporte neuf diagrammes. Ces diagrammes permettent d'obtenir une vue conceptuelle du système selon deux points de vue essentiels, l'un dynamique (i.e., diagramme de cas d'utilisation, diagramme de séquence) et l'autre statique (i.e., diagramme de classe) **[Roques 2001].**

Nous allons commencer par présenter la structure de la base *ArabicLDB*. Ensuite, nous présenterons la conception de son gestionnaire en spécifiant seulement les diagrammes de cas d'utilisation et de séquence.

II. Structure de la base

LMF est une norme de haut niveau qui spécifie **un méta-modèle (**quelquefois appelé **structure)**. Il s'agit de spécifier les meilleures pratiques du domaine et de couvrir la majorité des situations réelles. Ce méta-modèle est constitué des éléments importants et des liens d'association qui les relient. Néanmoins, la liste des attributs de chaque élément n'est pas décrite. Le document normatif spécifie que tel élément est destiné à tel usage et non à tel autre. Chaque élément doit être décoré par des couples attributs-valeurs (catégories de données complexes) à prendre obligatoirement à partir du registre de catégorie de données normalisées (ISO 12620).

Nous allons donc retenir le même méta-modèle proposé dans LMF et utiliser les catégories de données nécessaires pour la langue arabe en vue de créer le modèle approprié (voir Figure 51) à cette langue partant d'un ensemble de choix linguistique.

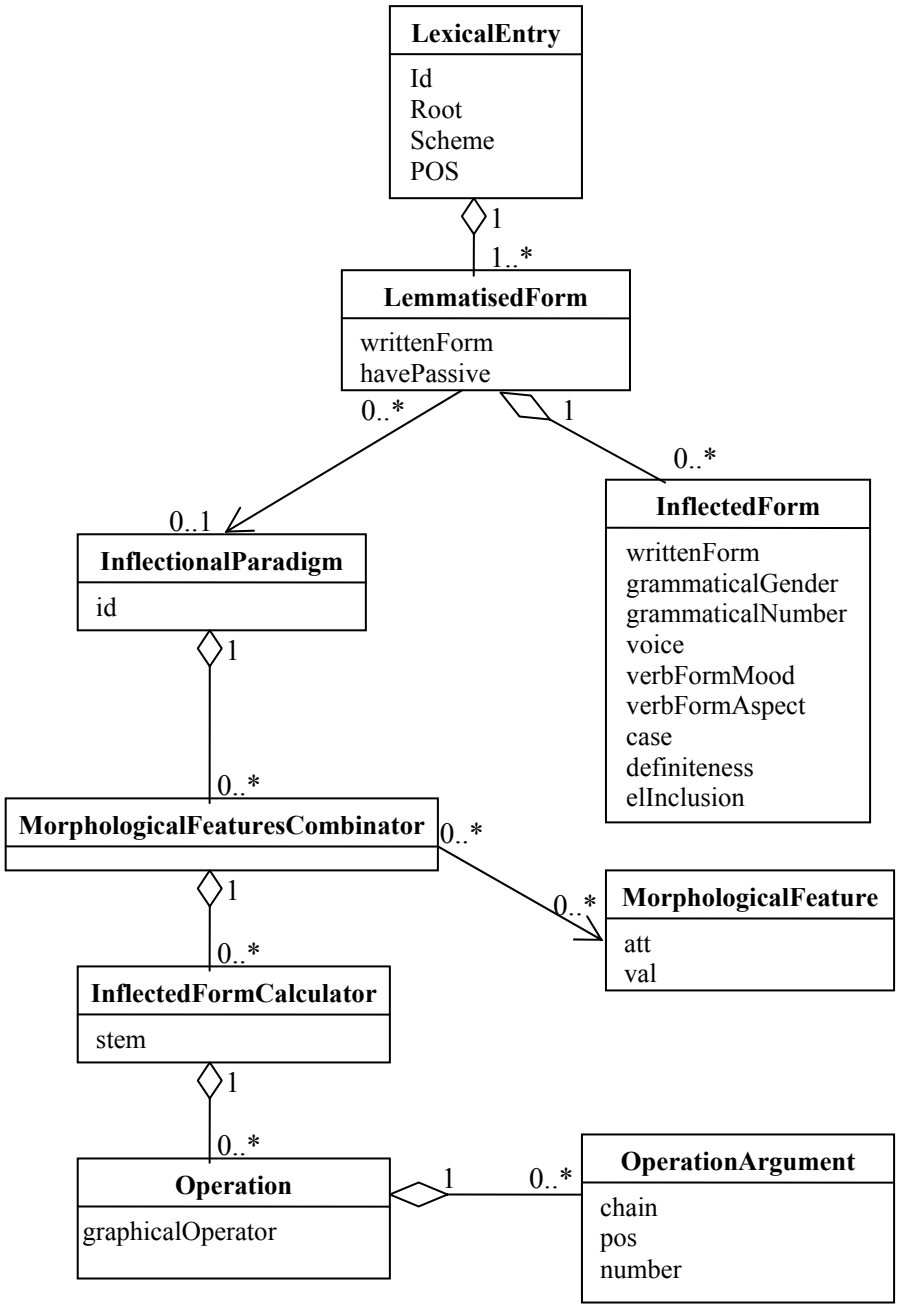


Figure 51 : Diagramme de classe décoré par les catégorie de données relatives à la langue arabe

Les différentes catégories de données seront représentées dans le DTD (voir annexe D) soit sous forme d'attribut, soit sous forme d'une balise vide nommée DC (DataCategory).

III. Le gestionnaire de la base

Le gestionnaire de la base est la partie dynamique de notre système qui permet l'alimentation et l'exploitation de *ArabicLDB*. Ce gestionnaire a deux principaux acteurs : l'administrateur et l'utilisateur. Nous allons élaborer une interface pour l'administrateur qui facilite l'alimentation de la base par les entrées lexicales, les formes fléchies pour les noms et les paradigmes de flexion pour les verbes. Ensuite, nous proposerons quelques

services (recherche, conjugueur) pour l'utilisateur, en vue d'exploiter le contenu de cette base.

Pour élaborer ce gestionnaire, nous allons visualiser, spécifier, construire et documenter les phases en amont de la réalisation, en précisant les diagrammes de cas d'utilisation et de séquence.

1. Diagramme de cas d'utilisation

Le diagramme de cas d'utilisation présenté dans la Figure 52 procède de la vue dynamique du système. Il décrit le fonctionnement du système vis-à-vis de l'administrateur et de l'utilisateur.

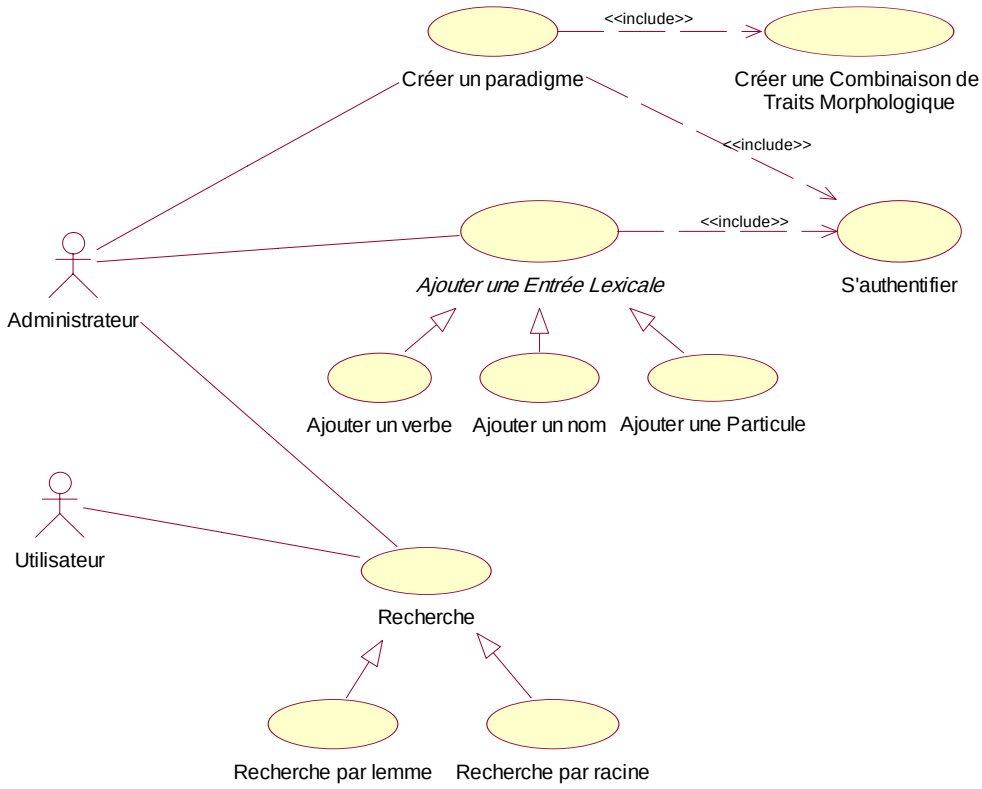


Figure 52 : Diagramme de cas d'utilisation de l'application

Nous signalons que le cas d'utilisation « *Ajout d'une Entrée Lexicale* » qui apparaît en italique, est un cas abstrait, car il ne s'instancie pas directement, mais uniquement par le biais de l'un de ses trois cas spécialisés. La même chose pour le cas d'utilisation « Recherche ». En plus, l'identification est une partie commune pour la plupart des cas d'utilisation, c'est la raison pour laquelle nous avons identifié un nouveau cas d'utilisation « S'authentifier » inclu dans les précédents.

1.1. S'authentifier

Titre : S'authentifier.

Résume : ce cas d'utilisation permet à un administrateur ou à un utilisateur d'accéder à l'un des autres cas d'utilisation.

Acteur : Administrateur ou Utilisateur (dans la description nous utilisons Utilisateur).

Description de scénarios

1. Ce cas d'utilisation commence quand un administrateur ou un utilisateur choisit d'utiliser ce système.	
2. L'utilisateur enregistre son nom et son mot de passe.	3. Le système vérifie ces informations et appelle le cas d'utilisation selon le nom.

1.2. Créer un paradigme

Titre : Créer un paradigme

Résume : ce cas d'utilisation permet à un administrateur d'ajouter un nouveau paradigme.

Acteur : Administrateur

Description de scénarios

1. Ce cas d'utilisation commence quand un administrateur choisit d'ajouter un paradigme.	
2. L'administrateur enregistre l'identifiant d'un nouveau paradigme.	
3. Après avoir enregistré l'identifiant, l'administrateur indique que l'identification est terminée.	4. Le système affiche toutes les combinaisons de traits morphologiques possibles.
5. L'administrateur choisit une combinaison de traits morphologiques.	6. Le système exécute le cas d'utilisation « Créer combinaison de traits morphologiques ». Ensuite, il enregistre cette combinaison.
7. Après avoir enregistré toutes les combinaisons, l'administrateur indique que l'ajout d'un paradigme est terminé.	8. Le système enregistre le nouveau paradigme dans la base XML.

1.3. Créer une combinaison de traits morphologiques

Titre : Créer une combinaison de traits morphologiques.

Résume : ce cas d'utilisation permet à un administrateur d'ajouter une nouvelle combinaison de traits morphologiques.

Acteur : Administrateur

Description de scénarios

1. Ce cas d'utilisation commence quand un administrateur choisit d'ajouter une combinaison de traits morphologiques.	
2. L'administrateur choisit l'opération et enregistre ses arguments (chaîne, position et nombre) selon le type de l'opération choisie.	
3. Après avoir préparé l'opération, l'administrateur indique que l'ajout d'une opération est terminé.	4. Le système stocke cette opération dans la liste et vide les champs de saisie pour une opération suivante.
5. Après avoir enregistré toutes les opérations, en répétant les points 2, 3 et 4 pour chaque opération, l'administrateur indique que le calcul d'une forme fléchie est terminé.	6. Le système stocke le calcul d'une forme fléchie dans la liste et vide la liste des opérations
7. Après avoir enregistré tous les calculs d'une forme fléchie, l'administrateur indique la création des formes fléchies.	8. Le système enregistre les calculs de la forme fléchie.

Enchaînement d'erreur

E1 : *L'administrateur enregistre une opération erronée.*

L'enchaînement E1 démarre au point 4 du scénario.

5. L'administrateur sélectionne l'opération erronée.

6. Après la sélection de toutes les opérations erronées, l'administrateur déclenche la suppression.

7. Le système supprime les opérations sélectionnées et affiche la nouvelle liste d'opération.

1.4. Ajouter un verbe

Titre : Ajouter un verbe

Résume : ce cas d'utilisation permet à un administrateur d'ajouter un verbe.

Acteur : Administrateur.

Description de scénarios

1. Ce cas d'utilisation commence quand un administrateur choisit d'ajouter un verbe.	
2. L'utilisateur enregistre la racine, le schème et le lemme. Il sélectionne son paradigme et il détermine les voix de ce verbe.	
3. Après avoir enregistré les informations de ce verbe, l'utilisateur indique la fin de l'enregistrement.	4. Le système ajoute cette entrée lexicale dans la base XML.

Enchaînements alternatifs

A1 : *nouvel ajout*

L'enchaînement A1 démarre au point 2 du scénario.

3. L'utilisateur déclenche un nouvel ajout.

4. Le système supprime toutes les informations précédentes.

1.5. Ajouter un nom

Titre : Ajouter un nom

Résume : ce cas d'utilisation permet à un administrateur d'ajouter un nom.

Acteur : Administrateur.

Description de scénarios

1. Ce cas d'utilisation commence quand un administrateur choisit d'ajouter un nom.	
2. L'administrateur enregistre la racine, le schème et le lemme. Il sélectionne son paradigme et sa sous catégorie grammaticale (POS). Il doit saisir les formes fléchies au singulier et au duel de ce nom.	

3. L'administrateur choisit le pluriel ou le pluriel brisé.	4. Le système affiche les zones de saisie selon le choix de l'utilisateur.
5. L'administrateur enregistre les formes fléchies aux pluriels.	
6. Après avoir enregistré les informations de ce nom, l'utilisateur indique la fin de l'enregistrement.	7. Le système ajoute cette entrée lexicale dans la base XML.

Enchaînements alternatifs

A1 : *nouvel ajout*

L'enchaînement A1 démarre aux points 2, 4, 5 et 7 du scénario.

3. L'utilisateur déclenche un nouvel ajout.

4. Le système supprime toutes les informations précédentes.

1.6. Ajouter une particule

Titre : Ajouter une particule.

Résume : ce cas d'utilisation permet à un administrateur d'ajouter une particule.

Acteur : Administrateur.

Description de scénarios

1. Ce cas d'utilisation commence quand un administrateur choisit d'ajouter une particule.	
2. L'administrateur enregistre le lemme. Il sélectionne sa sous catégorie grammaticale (POS).	
3. Après avoir enregistré les informations de ce nom, l'utilisateur indique la fin de l'enregistrement.	4. Le système ajoute cette entrée lexicale dans la base XML.

Enchaînements alternatifs

A1 : *nouvel ajout*

L'enchaînement A1 démarre aux points 2 et 4 du scénario.

3. L'utilisateur déclenche un nouvel ajout.

4. Le système supprime toutes les informations précédentes.

1.7. Recherche par lemme

Titre : Recherche par lemme

Résume : ce cas d'utilisation permet à un administrateur ou un utilisateur de chercher les informations d'un lemme et de générer ses différentes formes fléchies.

Acteur : Administrateur ou Utilisateur (dans la description nous utilisons Utilisateur).

Description de scénarios

1. Ce cas d'utilisation commence quand un administrateur ou un utilisateur choisit de chercher un lemme.	
2. L'utilisateur enregistre un lemme.	
3. L'utilisateur déclenche la recherche.	4. Le système affiche toutes les informations de ce lemme.
5. L'utilisateur déclenche la conjugaison de ce lemme si c'est un verbe ou un nom.	6. Le système exécute la conjugaison selon le type du lemme.
7. Après avoir la conjugaison, l'utilisateur peut enregistrer la représentation intentionnelle du lemme.	8. Le système enregistre les différentes formes fléchies du lemme, avec ces traits morphologiques dans un fichier XML.

Enchaînements alternatifs

A1 : *nouvelle recherche*

L'enchaînement A1 démarre aux points 4, 6, et 8 du scénario.

5. L'utilisateur déclenche une nouvelle recherche.

6. Le système supprime toutes les informations précédentes.

1.8. Recherche par racine

Titre : Recherche par racine

Résume : ce cas d'utilisation permet à un administrateur ou un utilisateur de chercher les formes dérivées verbales et nominales d'une racine.

Acteur : Administrateur ou Utilisateur (dans la description nous utilisons Utilisateur).

Description de scénarios

1. Ce cas d'utilisation commence quand un administrateur ou un utilisateur choisit de chercher une racine.	
2. L'utilisateur enregistre une racine.	

3. l'utilisateur déclenche la recherche.	4. Le système cherche toutes les entrées lexicales ayant cette racine et affiche leur schème et leur racine.

Enchaînements alternatifs

A1 : *nouvelle recherche*

L'enchaînement A1 démarre au point 4 du scénario.

5. L'utilisateur déclenche une nouvelle recherche.

6. Le système supprime toutes les informations précédentes.

2. Diagramme de séquence

Un diagramme de séquence établit une version temporelle des échanges entre objets. Les objets sont les colonnes du diagramme. Un envoi de message (un événement est représenté par une flèche entre deux objets. Une séquence est une suite d'envois de messages. Un envoi de messages induit une activité chez l'objet receveur.

A l'aide du diagramme de séquence, nous illustrons donc l'interaction entre objets en créant des liens entre ces objets et en associant des messages à ces liens. Le nom d'un message doit évoquer l'intention de l'objet appelé lors de l'interaction avec l'objet associé.

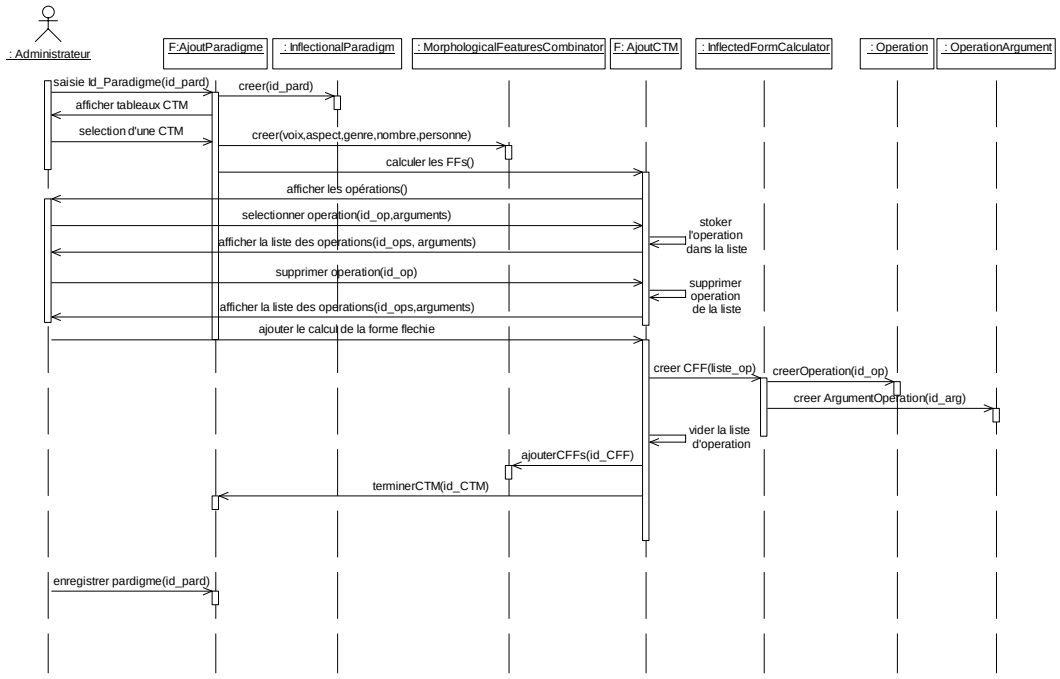


Figure 53 : Diagramme de séquence d'ajout d'un paradigme

Dans la Figure 53, l'administrateur est l'acteur qui peut ajouter un paradigme. Il utilise deux interfaces pour créer un paradigme, la première est pour saisir les propriétés du paradigme et pour sélectionner les combinaisons des traits morphologiques. La deuxième est générer suite à la sélection une combinaison des traits morphologique, elle permet de spécifier les opérations nécessaires pour calculer les formes fléchies de cette combinaison. Ces deux interfaces utilisent les différentes classes de l'extension de modes de flexion.

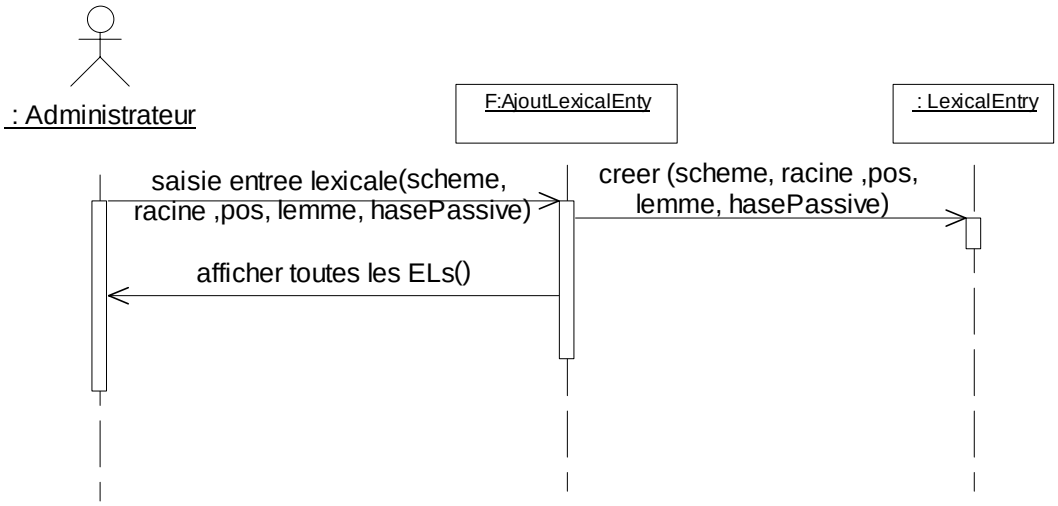


Figure 54 : Diagramme de séquence d'ajout d'une entrée lexicale

L'administrateur, dans la Figure 54, utilise l'interface AjoutLexicalEntry pour saisir les différentes caractéristiques d'une entrée lexicale. Sachant que ces caractéristiques dépendent de la catégorie grammaticale de l'entrée lexicale. En effet, cette interface comporte trois volets pour le nom, le verbe et la particule.

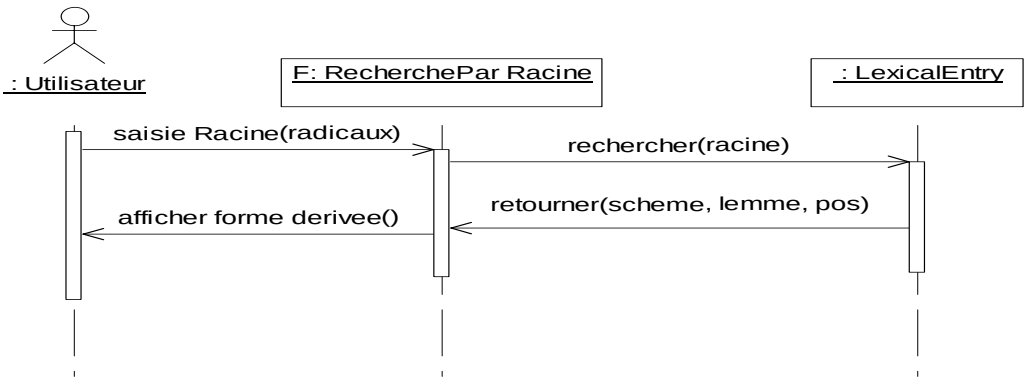


Figure 55 : Diagramme de séquence de recherche par racine

L'utilisateur déclenche la recherche par une racine, en utilisant l'interface RechercheParRacine qui manipule seulement la partie noyau de la base *ArabicLDB*, afin de chercher les formes dérivées ayant le même racine.

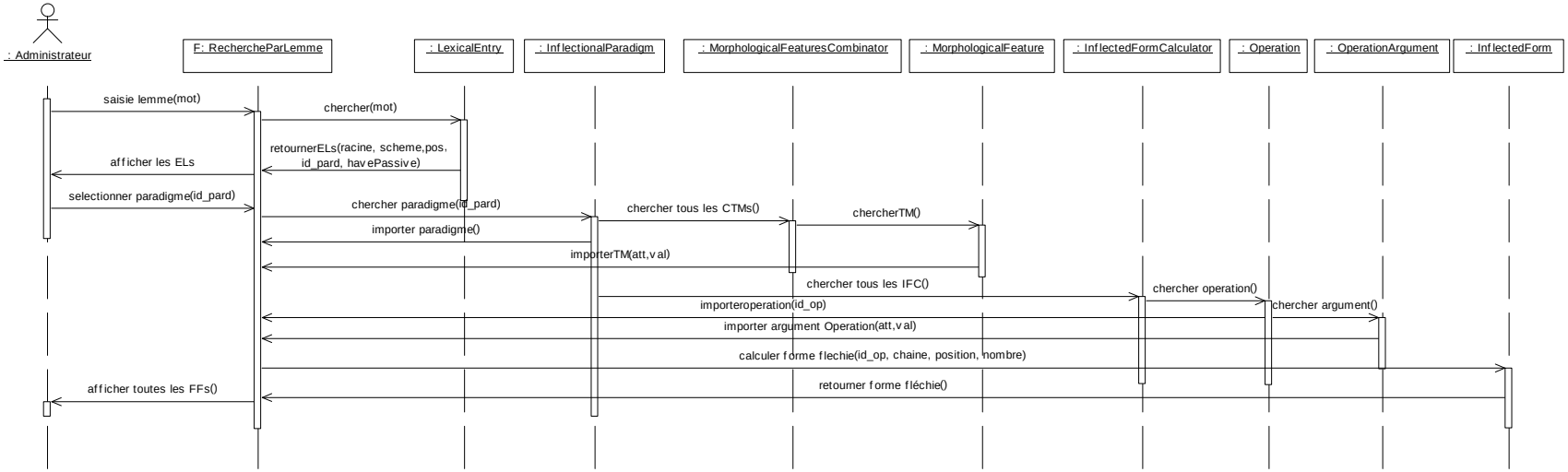


Figure 56 : Diagramme de séquence par lemme

Dans la Figure 56, l'utilisateur peut trouver les caractéristiques d'une entrée lexicale par le lemme. Ensuite à travers ces résultats, il peut déclencher un module de conjugaison qui permet de trouver toutes les formes fléchies du lemme. Ce module comporte plusieurs interactions internes entres les différentes classes des extensions (morphologique et mode de flexion).

IV. Conclusion

Dans ce chapitre, nous avons présenté la modélisation de la base *ArabicLDB* et de son gestionnaire qui va faciliter la gestion et l'exploitation de cette base. Nous rappelons que, nous nous sommes basés sur la révision 9 de LMF et par conséquent sur son DTD proposé (voir Annexe D). En effet, nous avons appliqué nos choix linguistiques sur le modèle de LMF, ensuite, nous avons conçu les interfaces nécessaires pour aider l'administrateur lors d'alimentation de la base, et pour offrir quelques services à l'utilisateur qui vise l'exploitation de *ArabicLDB*.

Partant du raffinement du méta-modèle de LMF et du DTD, nous allons aborder la phase d'implémentation de notre base lexicale.

CHAPITRE

6

Réalisation et jeux d’essai

I. Introduction

Ce chapitre a pour objet de présenter la réalisation de la base *ArabicLDB*. Pour réaliser cette base, nous avons utilisé le langage XML pour la description de son contenu et le langage de programmation JAVA pour la réalisation de son gestionnaire.

Il est à noter que notre système est divisé en deux parties dont chacune d'elle propose une vision différente. La première sera consacrée au point de vue de l'utilisateur qui vise l'exploitation de la base ArabicLDB. La seconde sera consacrée au point de vue de l'administrateur qui vise la gestion de la base.

En plus, nous avons mis en place un conjugueur qui exploite la base *ArabicLDB*, notamment sa composante de calcul des formes fléchies.

Nous allons commencer par présenter les langages et outils utilisés. Ensuite, nous décrirons l'architecture de notre système. Finalement, nous commencerons un jeu d'essai relatif au fonctionnement de notre système de gestion de la base *ArabicLDB* et au conjugueur associé.

II. Langages et outils utilisés

Pour réaliser ce travail, nous avons fait appel à deux technologies importantes dans le monde actuel de l'informatique à savoir XML et Java. La force de XML réside dans sa capacité de décrire n'importe quel domaine de données grâce à son extensibilité. Quant à Java, le choix de ce langage pour mon projet est dû à sa caractéristique la plus connue, à savoir la portabilité. Ce choix permettra à l'ensemble du projet de tourner sur de multiples plateformes. Cette caractéristique est encore accentuée par l'utilisation du XML.

1. XML

XML (eXtensible Markup Language) est un langage de balises qui peut être considéré comme un métalangage permettant de définir d'autres langages, autrement dit, il permet de définir de nouvelles balises (markup). Il découle d'un langage défini en 1986, le SGML (Standard Generalized Markup Language). Ensuite, il est standardisé par W3C en 10/02/1998. XML est une norme puissante et acceptée par la majorité. Il permet d'archiver et de communiquer des informations sur les objets. Il est en train de devenir le langage universel d'échange des documents.

La puissance de XML vient de la liberté de définir des balises et également leur structure selon le besoin de l'application. Pour définir ces balises, nous allons utiliser la DTD qui nous permet de préciser les balises à utiliser dans l'application.

La DTD de notre base est celle de la norme LMF proposée dans la version 9 **[Francopoulo et al, 2006a]**. Nous présenterons cette DTD dans l'annexe D. Ci-dessous nous présentons une entrée lexicale de la base ArabicLDB :

```
<lexicalEntry>
    <DC att="root" val="ك ت ب " />
    <DC att="scheme" val="فَعَلَ" />
    <DC att="pos" val="verb" />
    <lemmatisedForm paradigm="asKataba">
        <DC att="wordForm" val="كَتَبَ" />
        <DC att="havePassive" val="yes" />
    </lemmatisedForm>
</lexicalEntry>
```

Figure 57 : L'entrée lexicale "كَتَبَ " en XML

2. JAVA

Java est un langage de programmation de quatrième génération développé par Sun. Sa popularité évolue grâce à une caractéristique majeure, sa portabilité.

Dans le cadre de développement d'une interface de gestion et d'exploitation d'une base de données XML, les bibliothèques Swing et JDOM sont les plus importantes. La bibliothèque Swing propose une série d'objets graphiques tels que des fenêtres, des boutons, etc. Elle est la base de toute l'interface que nous avons développée. La bibliothèque JDOM offre des classes spécialisées pour un document XML tels que Document, Element, etc. Elle est la base de tous nos accès à la base soit en mode écriture soit en mode lecture.

3. JDOM

JDOM est une API évoluée du langage Java développée indépendamment de Sun Microsystems. Elle permet de manipuler des donnés XML plus simplement qu'avec les API classiques. Son utilisation est pratique pour tout développeur Java et repose sur les API XML de Sun.

JDOM utilise des collections SAX pour parser les fichiers XML. En outre, JDOM utilise DOM pour manipuler les éléments d'un Document Object Model spécifique (créé grâce à un constructeur basé sur SAX). Ainsi, JDOM nous permet de construire des documents, de naviguer dans leur structure, d'ajouter, de modifier, ou de supprimer leur contenu **[Cynober, 2005]**.

4. Oxygen XML Editor

oXygen (http://www.oxygenxml.com/) est un éditeur simple à utiliser et élégant qui supporte les dernières technologies et standards XML. Il s'agit d'un logiciel offrant un certain nombre de fonctionnalité facilitant la saisie de code XML. oXygen peut valider des documents XML qui utilisent une DTD ou un XML Schema. Nous allons utiliser cet outil pour valider notre base en respectant la DTD proposée par LMF (voir annexe D).

III. Architecture du système

Notre environnement présenté dans la Figure 58 est conçu pour faciliter la génération et la manipulation de la base ArabicLDB, par les modules suivants :

- **Module d'acquisition d'une entrée lexicale** : il englobe trois méthodes différentes selon la catégorie grammaticale (nom, verbe et particule). Il accède à l'extension morphologique de ArabicLDB en mode écriture pour l'ajout d'une nouvelle entrée.
- **Module d'acquisition d'un paradigme de flexion** : il englobe deux méthodes différentes, une pour les verbes et une autre pour les noms (travail en cours). La différence se présente au niveau de choix des traits morphologiques. Il accède à l'extension pragmatique de ArabicLDB en mode écriture pour l'ajout d'un nouveau paradigme.
- **Module de recherche par lemme** : il accède à l'extension morphologique de ArabicLDB en mode lecture pour importer l'entrée lexicale correspondante. Ensuite, si c'est un verbe, il appelle le module de génération des formes fléchies, il envoie le lemme et l'identifiant du paradigme de l'entrée recherchée. Si c'est un nom ou particule, il appelle directement le module de l'affichage des formes fléchies
- **Module de génération des formes fléchies** : il accède par l'identifiant d'un paradigme à l'extension des modes de flexion en mode lecture pour importer le paradigme. Ensuite, il calcule toutes les formes fléchies à partir du lemme, en stockant ces résultats dans un fichier XML temporel. Enfin, il appelle le module d'affichage des formes fléchies.
- **Module de recherche par racine** : il accède à l'extension morphologique de ArabicLDB en mode lecture pour importer l'ensemble des entrées lexicales qui ont cette racine. Ensuite, il appelle directement le module de l'affichage des formes dérivées en envoyant les résultats obtenus.

- **Module d'affichage des formes fléchies** : il fonctionne selon la catégorie grammaticale, il accède au fichier XML temporel pour afficher les formes fléchies selon leurs traits morphologiques.
- **Module d'affichage des formes dérivées** : il permet l'affichage des formes dérivées d'une racine avec leur schème.

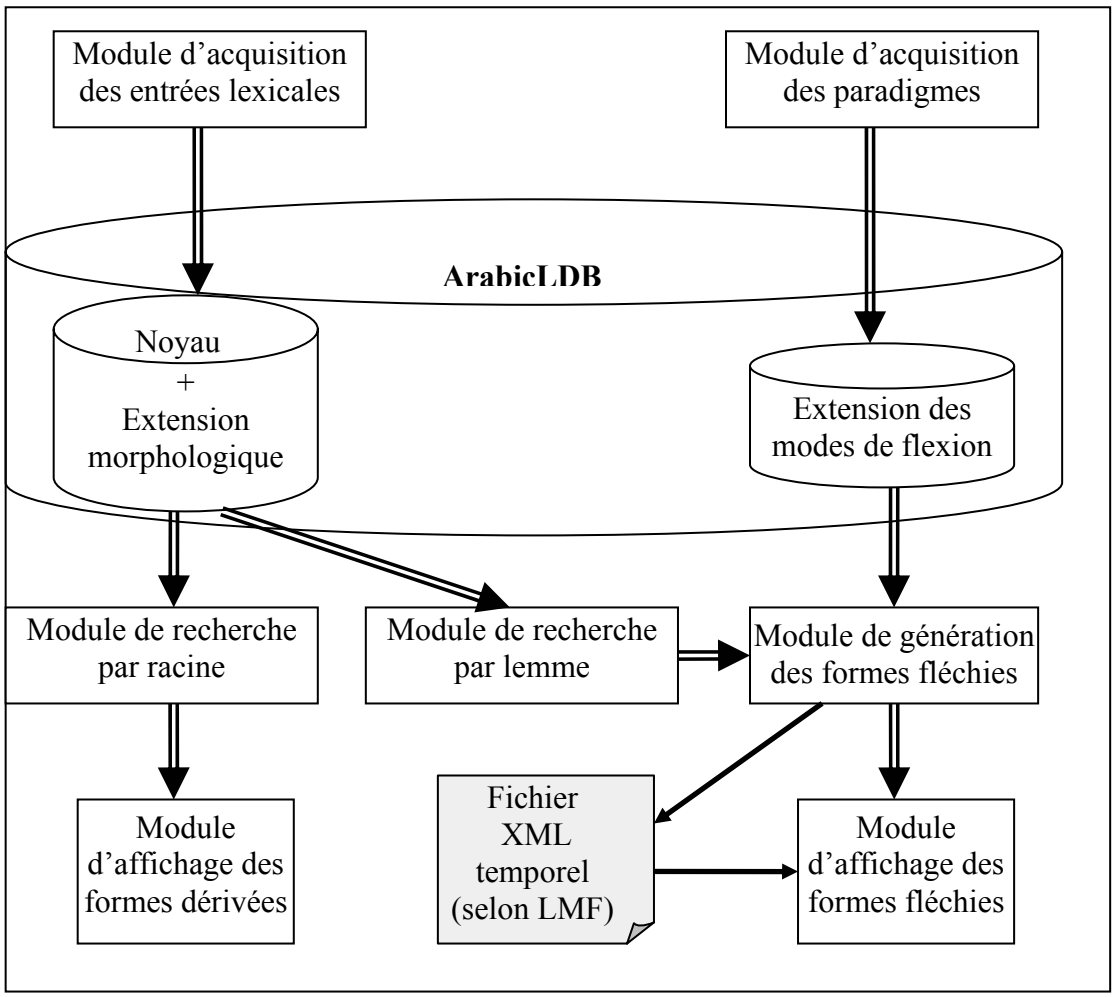


Figure 58 : Architecture de l'environnement de gestion et d'exploitation de la base ArabicLDB

Le fichier XML temporel peut être enregistré, après la demande de l'utilisateur. Nous avons présenté un seul exemple d'exploitation de la base ArabicLDB qui est la recherche par lemme ou par racine, mais il y a plusieurs méthodes d'exploitation qui sont en cours d'élaboration au sein de notre équipe. Ces travaux visent l'exploitation maximale de la base selon les besoins des applications de TALN.

IV. Jeux d'essai

L'interface du programme est constituée d'éléments de la bibliothèque Swing et d'éléments dérivés. La plupart des éléments de Swing sont dérivés de la classe

JComponent. Nous avons choisi de présenter seulement le menu principal de l'administrateur qui englobe toutes les commandes possibles : ajout de paradigme, ajout des entrées lexicales et les différents types de recherche.

4.1. Menu

Le menu de l'administrateur présenté dans la Figure 59 comporte les trois fonctions principales, mais pour celui de l'utilisateur, les fonctions d'ajout sont invisibles.

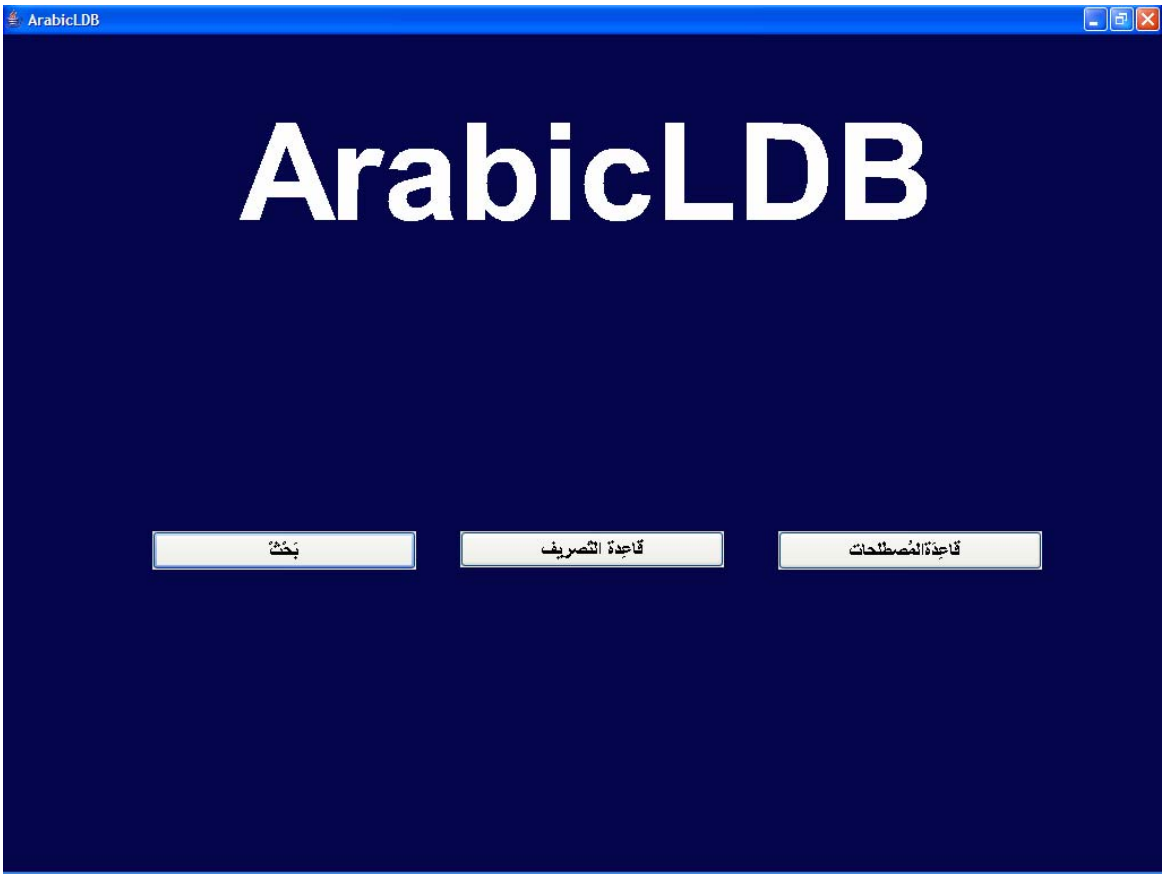


Figure 59 : Menu principal de ArabicLDB

4.2. Acquisition d'un paradigme

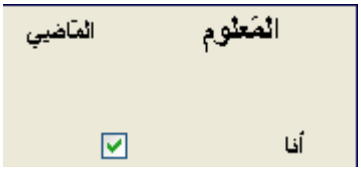
Au cours de chargement de cette page, il y a préparation des combinaisons de traits morphologiques (genre, nombre, aspect, voix, mode et personne) qui seront sélectionnées par l'administrateur pour les ajouter au nouveau paradigme. Chaque sélection va générer une nouvelle combinaison de traits morphologiques et appeler l'interface de génération des formes fléchies présentée dans la Figure 60.

Il y a quatre boutons de commande dans cette interface. Le premier « أنْشِئ القَاعِدَة » permet la création d'un élément XML *InflactionalParadigm*, le changement de la propriété du bouton d'enregistrement pour qu'il soit accessible et l'affichage des combinaisons de traits morphologiques sous forme de tableaux comme c'est indiqué ci-dessous. Une fois

l'opération terminée ce premier bouton sera invisible. Après la sélection et la création des combinaisons de traits morphologiques nécessaires pour ce paradigme, le deuxième bouton « سَجِّل القَاعِدة » permet l'accès à la base en mode écriture pour ajouter ce nouveau paradigme. Ensuite, vient le rôle du troisième bouton « قَاعِدة جَدِيدة » qui consiste à la réinitialisation de l'interface. Un quatrième bouton sert à l'appel du menu principal.

Figure 60 : Création des nouveaux paradigmes

L'interface d'ajout des formes fléchies est appelée à chaque nouvelle sélection d'une combinaison de traits morphologiques. Elle est présentée dans la Figure 61.



Dans cette interface figure la liste des opérations proposées par LMF. Il faut en sélectionner une opération et en spécifier les arguments nécessaires. Ensuite le bouton « Ajouter opération » permet d'ajouter l'opération sélectionnée dans la liste à droite. En cas d'erreur, on peut sélectionner les opérations erronées et utiliser le bouton « Supprimer opérations » qui supprime ces opérations.

Après la préparation de toutes les opérations nécessaires pour le calcul d'une forme fléchie, on utilise le bouton « Valider forme fléchie » qui permet la création d'un élément XML *InflectedFormCalculator* et vide la liste des opérations pour préparer l'interface pour une autre forme fléchie si cette combinaison de traits morphologiques a plusieurs formes fléchies. En parallèle, le bouton « Terminer CTM » sera accessible. Il permet d'ajouter ces *InflectedFormCalculator* à la combinaison de traits morphologiques et de retourner à l'interface précédente pour terminer le paradigme.

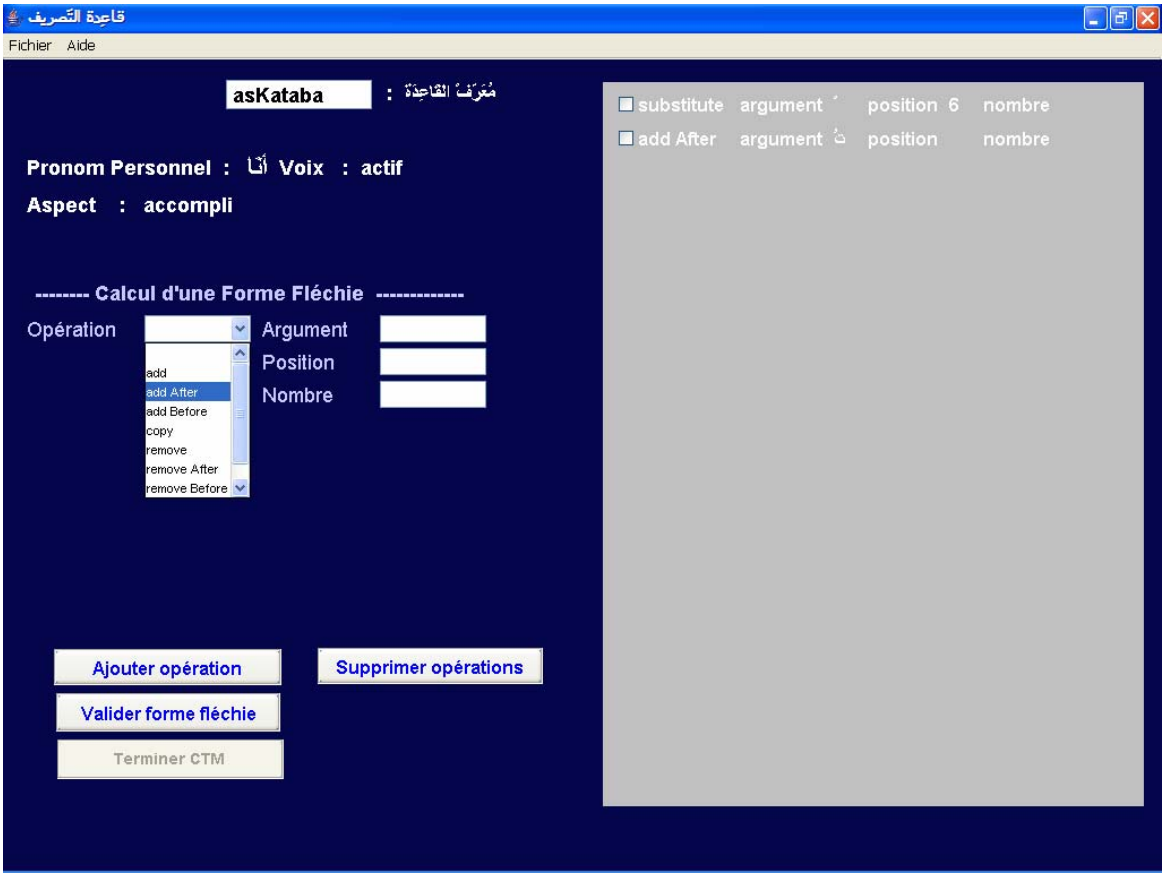


Figure 61 : Création des formes fléchies

Cette interface ne nous permet pas de passer au menu principal pour assurer la continuité des tâches.

4.3. Acquisition d'un nom

La première interface permet de choisir le type du nom. Ce type dépend des formes fléchies que peuvent avoir le nom, nous avons cinq types. Dans la Figure 62, la sélection

de « اِسْم مُعْرَب » rend les choix de même niveau inaccessible et ses sous choix seront accessibles pour en choisir un seul. Selon le choix, le bouton de validation génère les champs nécessaires pour ce type de nom.

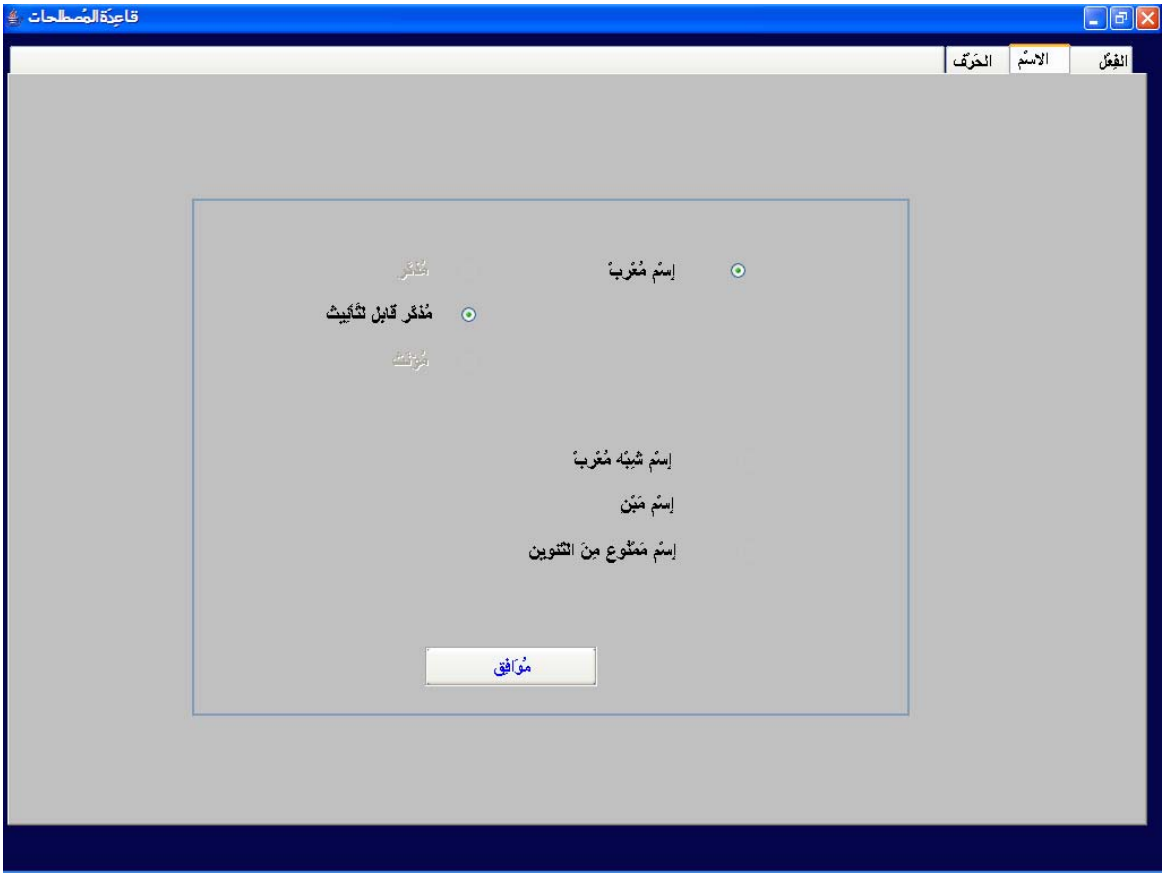


Figure 62 : Interface de sélection d'un type de nom

Par exemple, la Figure 63 présente l'interface des noms conjugables qui a deux grandes parties. La première comporte les champs nécessaires pour une entrée lexicale comme le verbe, mais les paradigmes pour les noms ne sont pas encore préparés et la propriété de passif n'est pas valable pour les noms. La deuxième est nécessaire pour saisir les formes fléchies de cette entrée.

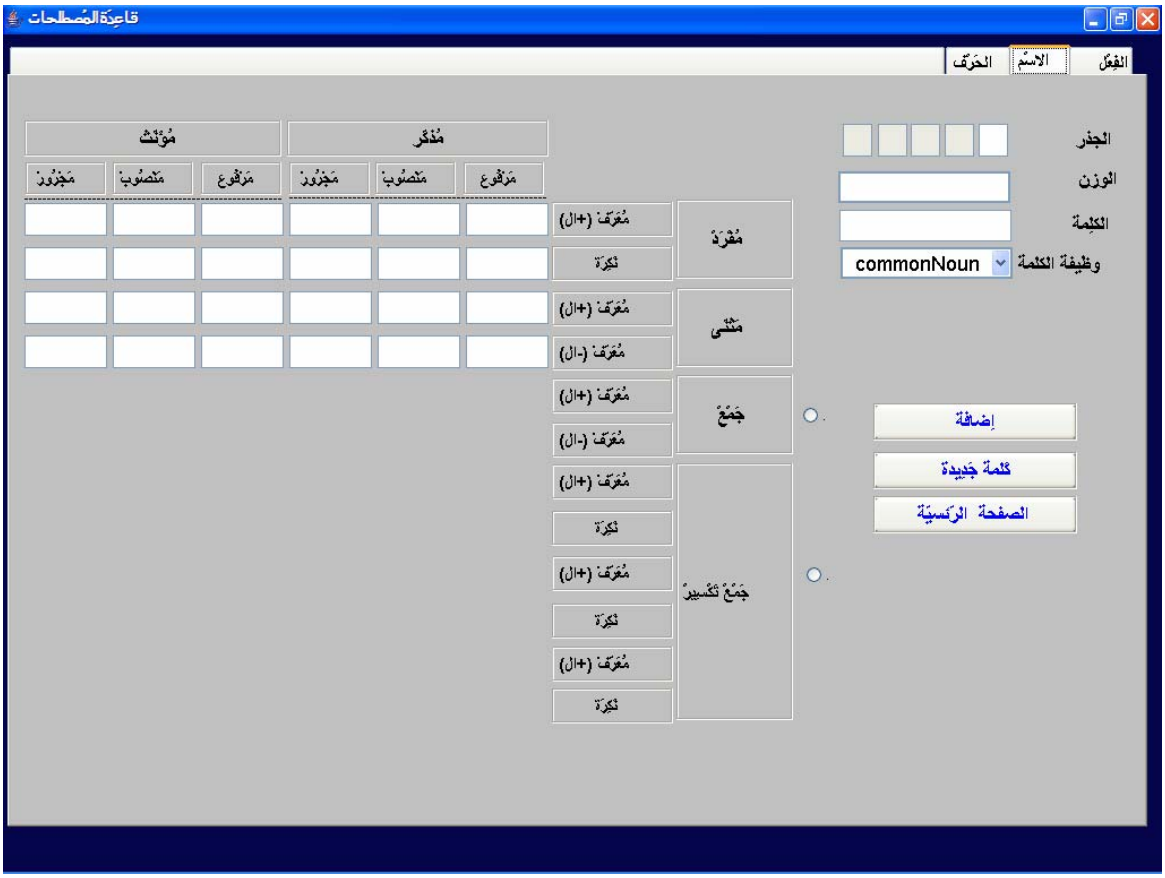


Figure 63 : Création des nouveau noms

4.4. Acquisition d'un verbe

Dans le chargement de cette page, il y a un accès à la base des paradigmes pour importer les identifiants de tous les paradigmes dans une lite box, ceci facilite la sélection du paradigme convenable pour le nouveau verbe.

Nous avons contrôlé les étapes de saisie des données :

- Au niveau de la racine, chaque consonne est dans une case à part. Lors de la saisie, il y a une vérification automatique qui n'accepte que les consonnes arabes.

- Au niveau du schème, il y a une liste limitée de schème pour les verbes (20 au total) qui dépend du nombre des radicaux par exemple si la racine est quadrilitère, il faut cacher les schèmes trilitères comme فَعَلَ et فَعُلَ .

▪ Au niveau du lemme, il y a un processus de génération du lemme qui se fait automatiquement à partir de la racine et du schème. Ce processus permet de remplacer les consonnes de la racine du schème par les consonnes de la racine du nouveau verbe. Mais il y a toujours des cas particuliers tels que :

- Pour les verbes trilitères, si les deux derniers radicaux sont identiques, il faut en garder un seul avec le signe de *šadda* et ajouter la voyelle *fatha* ـَـ.
- La graphie de *hamza* sera sur *alif* : "أ" quelle que soit sa voyelle, si elle est la première consonne de la racine.
- La graphie de *hamza* dépend de la voyelle précédente : si elle se trouve au milieu et que sa voyelle est muette (*sukûn* ـْـ), si elle est à la fin et que la voyelle précédente n'est pas muette.
- La graphie de *hamza* dépend de sa voyelle, si elle est au milieu et que sa voyelle n'est pas muette.
- La graphie de *hamza* sera sur la ligne "ء", si elle est à la fin et que la voyelle précédente est *sukûn* ـْـ ou si elle est précédée par une voyelle longue.
- La graphie de *hamza* dépend de sa voyelle et de la voyelle précédente, si elle est à la fin et que les deux voyelles ne sont pas muettes. Dans ce cas si une de ces deux voyelles est *kasra* ـِـ, elle domine les autres. S'il y a la voyelle *ḍamma* ـُـ, elle domine la voyelle *fatha* ـَـ. En plus, l'influence de *kasra* ـِـ, engendre la graphie suivante "ئ", l'influence de *ḍamma* ـُـ, engendre la graphie suivante "ؤ" et l'influence de *fatha* ـَـ, engendre la graphie suivante "أ".

Il est à noter qu'il y a d'autres cas particuliers qui nécessitent la collaboration d'un linguiste avec l'administrateur. Citons en l'occurrence la présence des lettres défectueuses dans la racine.

Il y a quatre boutons de commande, le premier «إضافة» permet l'ouverture de la base en mode écriture pour ajouter une nouvelle entrée lexicale, ensuite il affiche le contenu de base à gauche. Le deuxième « كلمة جديدة » permet de réinitialiser les champs de saisie pour une nouvelle entrée lexicale. Le troisième « المداخل المعجميّة » permet d'accéder à la base en mode lecture pour afficher son contenu, à gauche de l'interface. Enfin, le quatrième « الصّفحة الرّئسيّة » permet le retour au menu principal.

L'organisation de cette interface est présentée dans la Figure 64.

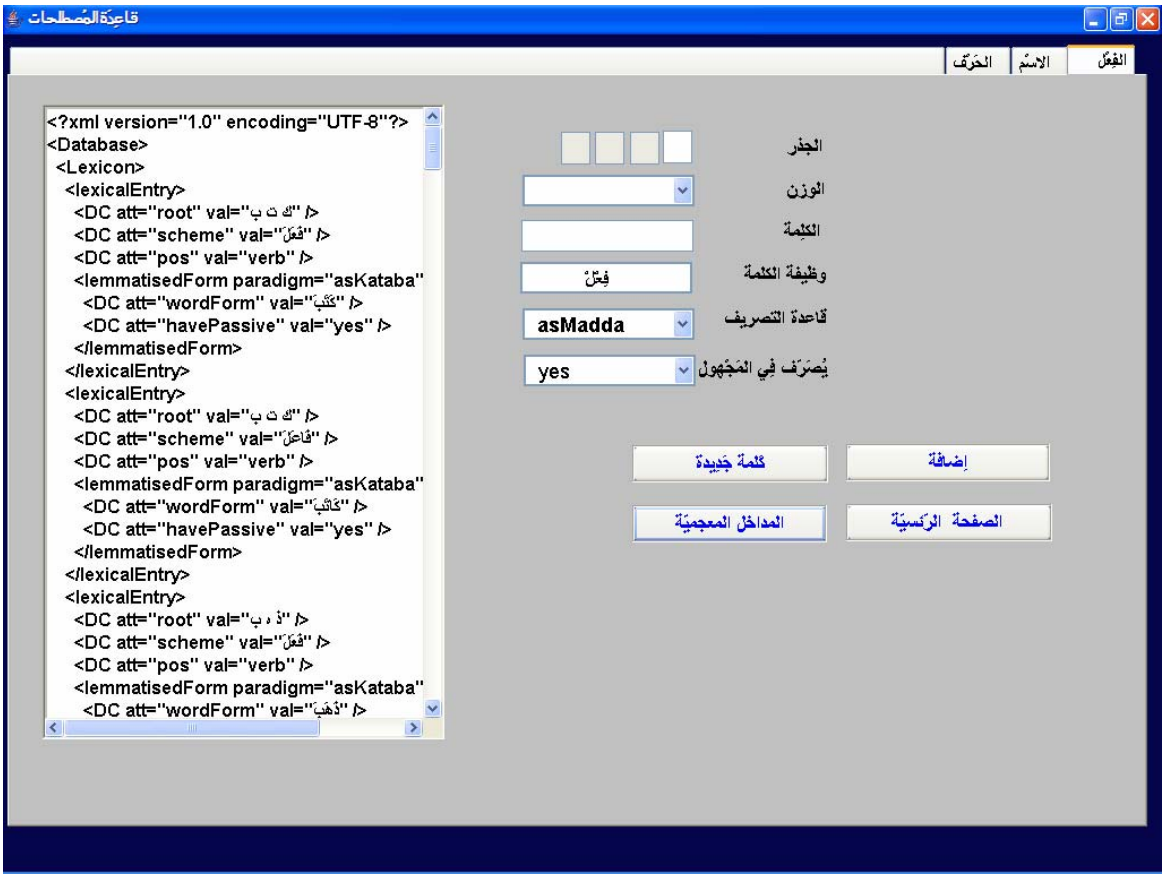


Figure 64 : Création des nouveaux verbes

4.5. Acquisition d'une particule

Cette interface est simple, elle est présentée dans la Figure 65. Elle consiste seulement à spécifier la particule et à sélectionner sa catégorie grammaticale. Le bouton d'ajout permet l'accès à la base en mode écriture pour ajouter cette nouvelle entrée lexicale. Comme dans les autres interfaces, il y a un bouton de réinitialisation et un autre pour l'appel du menu.

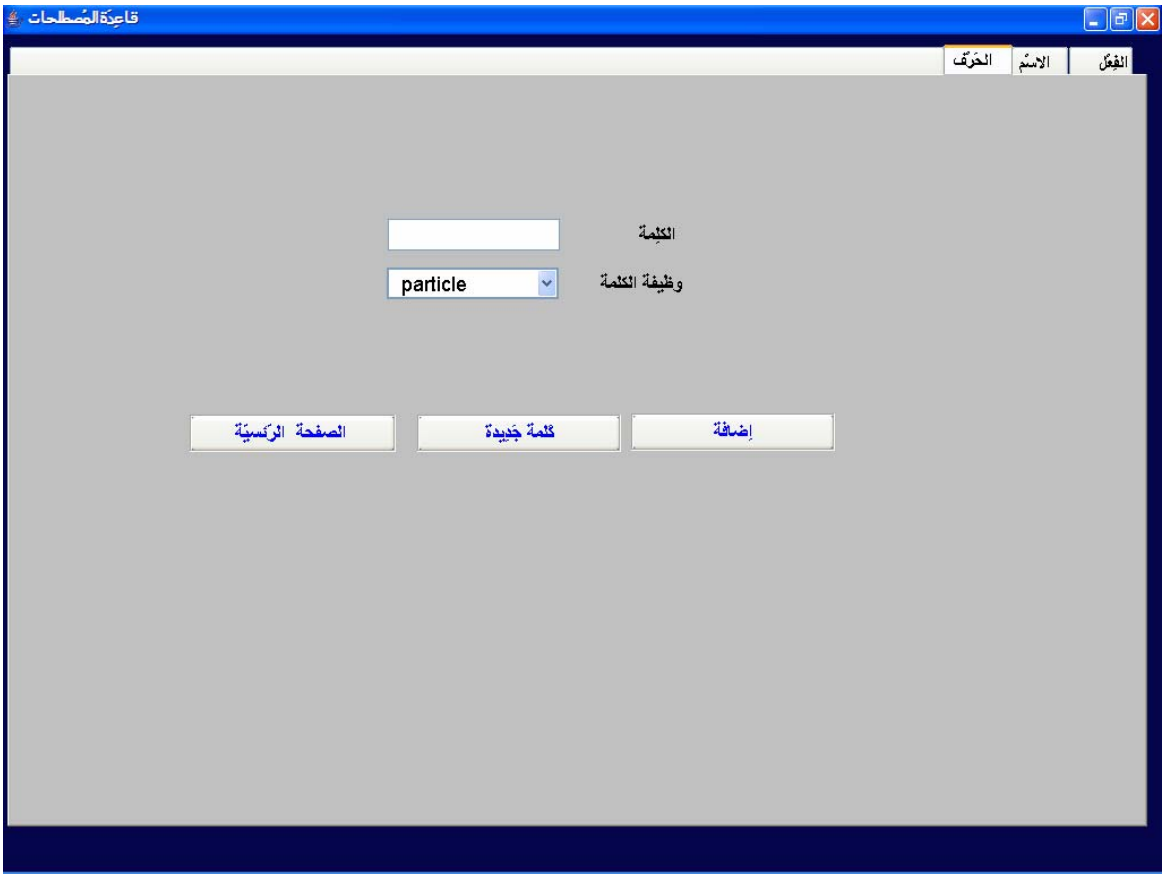


Figure 65 : Création des nouveaux particules

4.6. Recherche par racine

Cette interface comporte cinq zones de saisie pour la racine et trois boutons : le premier lance le processus de la recherche, le second réinitialise l'interface pour une nouvelle recherche et le troisième appelle le menu. Le processus de recherche accède à la base en mode de lecture pour importer toutes les entrées lexicales ayant en commun cette racine. Ces entrées peuvent être soit verbale soit nominale, selon sa catégorie, on classe ces entrées et on affiche leur schème et leur lemme comme l'exemple de la Figure 66.

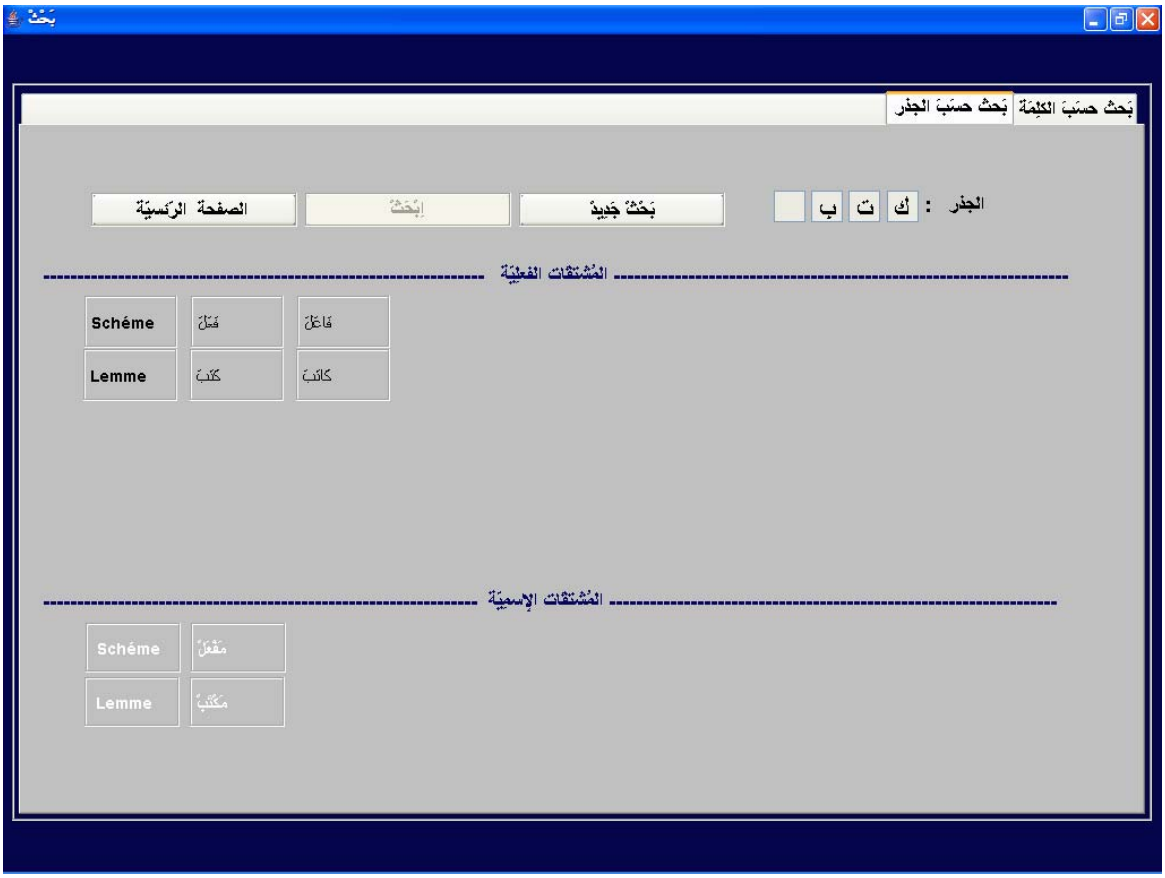


Figure 66 : Recherche par racine de "ك ت ب"

4.7. Recherche par lemme

Cette interface comporte une zone de saisie et trois boutons : le premier lance le processus de la recherche, le second réinitialise l'interface pour une nouvelle recherche et le troisième appelle le menu. Le processus de recherche accède à la base en mode de lecture pour importer toutes les entrées lexicales ayant en commun ce lemme. Ensuite, il appelle une fonction pour afficher les informations de chaque entrée lexicale. La Figure 67 est le résultat de la recherche du lemme "كَتَبَ".

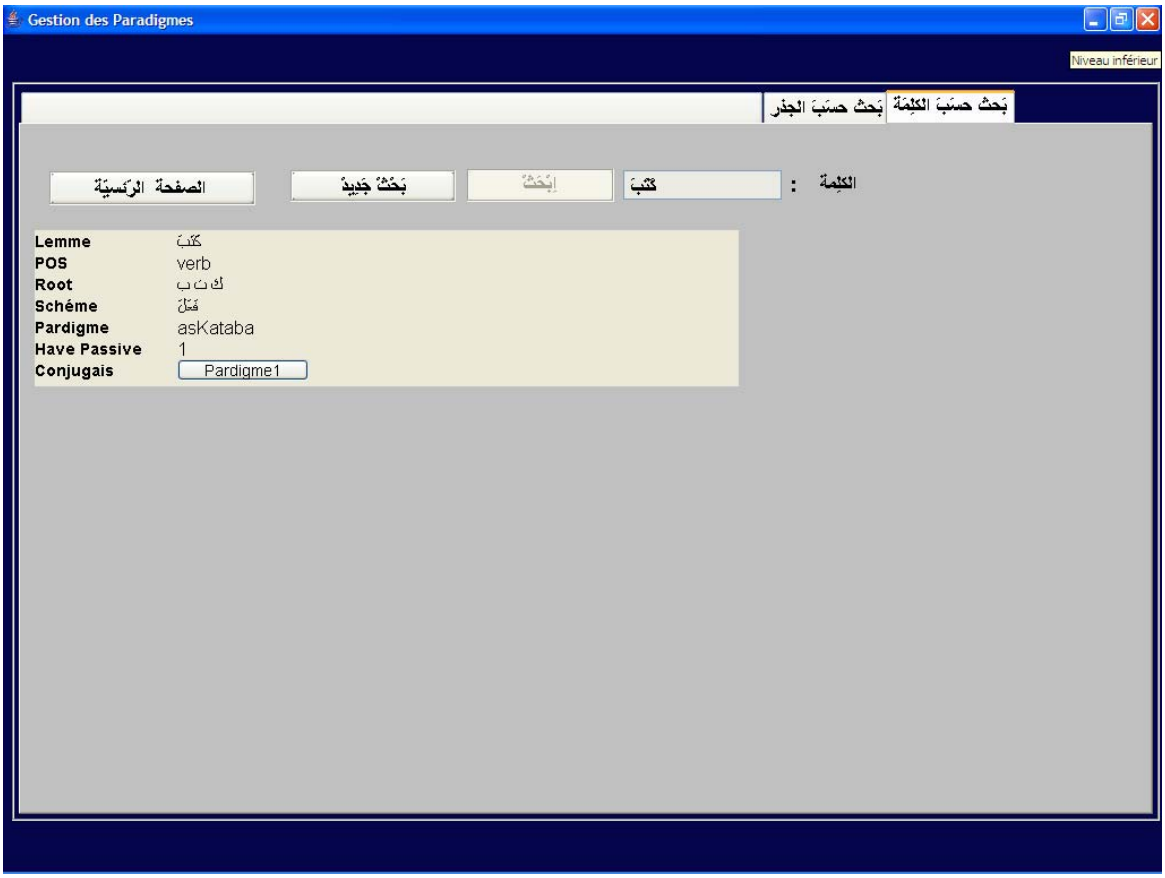


Figure 67 : Recherche du lemme"كَتَبَ"

4.8. Outil de conjugaison

Ce conjugueur se base sur les résultats de la recherche par lemme. En effet, si la catégorie grammaticale du lemme recherché est un verbe, le bouton de conjugaison génère les tableaux de conjugaison de ce lemme selon un processus de conjugaison.

Le processus de conjugaison accède à la base des paradigmes avec son identifiant pour l'importer. Ensuite, il utilise les méthodes offertes par l'API JDOM pour accéder à chaque *MorphologicalFeaturesCombinator* qui est caractérisé par des traits morphologiques et des *InflectedFormCalculator* qui englobent des opérations. Chaque opération a des arguments qui sont nécessaires à l'appel de la fonction du calcul d'une forme fléchie qui utilise des méthodes de traitement des chaînes de caractères comme *substring*, *getChars*, etc.

La Figure 68 est le résultat de la conjugaison du verbe "كَتَبَ".

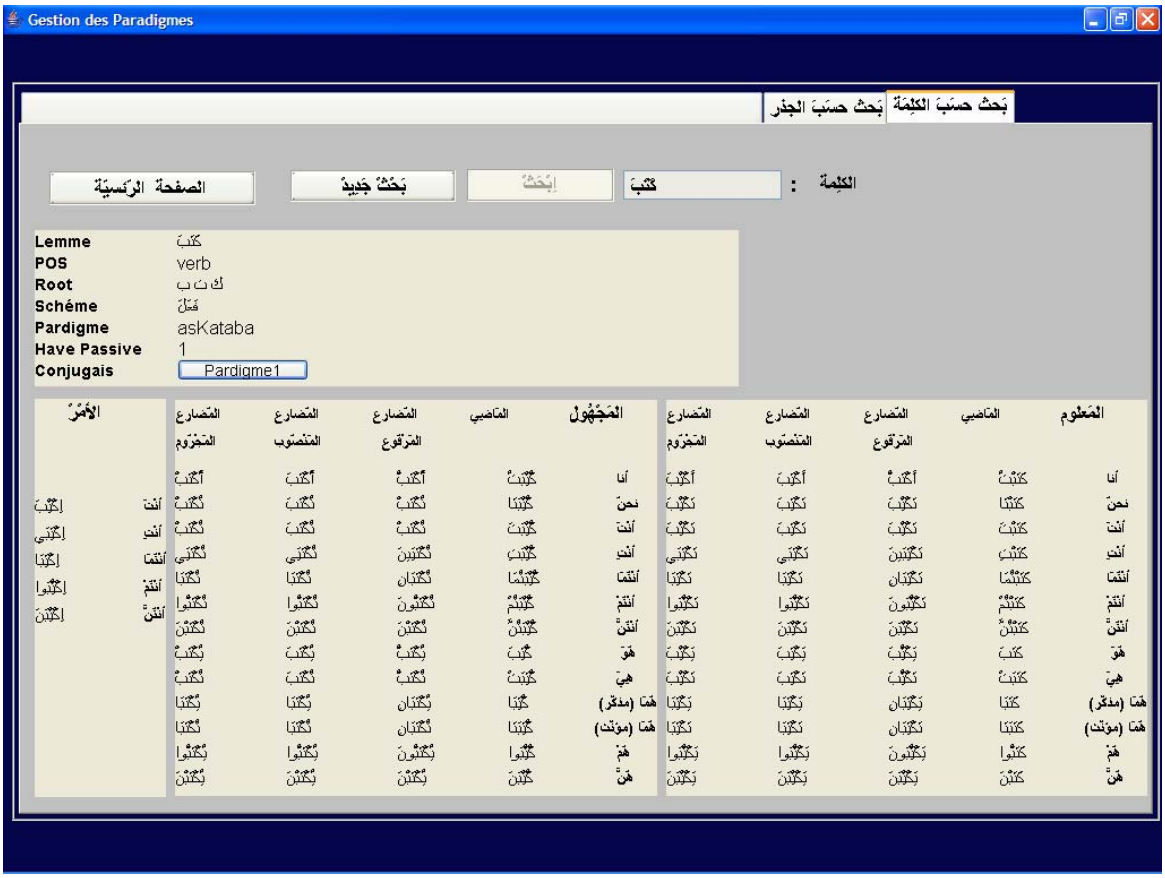


Figure 68 : Tableau de conjugaison de " كَتَبَ "

V. Conclusion

ArabicLDB est le nouveau né des bases lexicales arabes. Cette base est conforme à la future norme ISO 24613 dans sa version actuelle (révision 9 de Mars 2006). La réalisation de cette base est effectuée en XML avec un gestionnaire en JAVA.

Par ailleurs, *ArabicLDB* est dotée, dans sa version actuelle, d'un conjugueur des verbes sur un processus de calcul des formes fléchies. En effet, la représentation des verbes dans *ArabicLDB* est intentionnelle alors qu'elle extensionnelle pour les noms.

Actuellement, la base *ArabicLDB* est en cours de remplissage. Elle comporte déjà tous les paradigmes de flexion verbales (259 formes) et un bon nombre d'entrées lexicales variées (verbes, noms et particules)

Conclusion générale et perspectives

Le présent travail de recherche est effectué dans le cadre d'un projet de coopération entre notre laboratoire MIRACL, l'unité de recherche LSCA et le laboratoire Loria/INRIA. Son objectif a été dans un premier temps d'étudier les possibilités d'appliquer LMF, future norme ISO 24613 de représentation des ressources lexicale, sur la langue arabe. Dans un deuxième temps, notre objectif a été aussi de concrétiser cette étude par la conception et la mise en place d'une base lexicale de l'arabe qui soit conforme à cette future norme. Cette base, baptisée ArabicLDB (Arabic Lexical Data Base), sera utile pour les applications du traitement automatique de l'arabe.

Notre travail s'est limité au niveau morphologique. Ainsi, nous avons traité, outre le noyau de LMF, l'extension morphologique et celle des modes de flexion.

En ce qui concerne l'application de LMF sur la morphologie de l'arabe, nous avons trouvé que les parties étudiées (noyau, extension morphologique et extension des modes de flexion) sont applicable moyennant certaines adaptations (ou choix linguistiques) qui sont illustrées dans la réalisation de la base ArabicLDB.

Dans l'élaboration de cette base, nous avons donc modélisé le noyau, une extension morphologique ainsi qu'une extension des paradigmes de flexion verbale. Nous avons généré 259 paradigmes selon une approche combinatoire des classes des racines des verbes avec les différents schèmes associés. Ces paradigmes couvrent plus que 16 000 entrées lexicales verbales. Ainsi, nous avons suivi la représentation intentionnelle qui réduit la taille de la base du côté des verbes. Cependant, l'absence d'études suffisantes sur les flexions des noms, nous a conduit à une représentation extensionnelle pour ces derniers.

En ce qui concerne la conformité à LMF, nous pouvons affirmer que la base *ArabicLDB* est bien conforme. En effet, l'utilisation de l'outil Oxygen (http://www.oxygenxml.com/) a permis de valider le contenu de la base par rapport à la DTD fournie dans la dernière révision de LMF (révision 9 au 15-03-2006).

En vue de créer la base *ArabicLDB*, nous avons développé les outils nécessaires pour l'acquisition et la consultation. Entre autres, nous avons implémenté un conjugueur qui utilise les paradigmes de flexion lors du calcul des formes fléchies ainsi que deux modules de recherche : la recherche par lemme qui offre toutes les informations liées à celui-ci et la recherche par racine qui permet d'accéder à la base pour importer toutes les entrées qui représentent les formes dérivées, soit nominales ou verbales, de la racine. Le volet du développement d'outils d'exploitation de la base *ArabicLDB* pour le compte des applications du TALN est déjà abordé. En effet, dans le cadre du même projet de

coopération, un travail de Master a permis d'assurer cette exploitation en adoptant une approche orientée services (i.e., Services Web) qui vient consolider la réalisation de notre base **[Chaari et al, 2006]**.

En ce qui concerne la classification proposée par la norme ISO 12620, nous avons proposé d'ajouter certaines catégories, en particulier des sous-catégories spécifiques à la langue arabe comme *nom de temps*, *masdar* et *pronom allusif*.

D'un autre côté, le nombre de paradigmes verbaux peut être réduit pour passer de 259 (selon notre réalisation) à une vingtaine seulement en considérant des règles d'ordre phonologique. En effet, la langue arabe est caractérisée par un lien très fort entre ses différents niveaux de traitement notamment entre les niveaux phonologique et morphologique.

Dans le présent travail, nous avons traité uniquement les verbes conjugables « الأفعَال المُتصرّفة » dont le nombre dépasse 16000 verbes, mais il reste à étudier les verbes semi conjugables « أفعال نَاقِصة التّصريف » et les verbes non conjugables « أفعَال جامدة » **[El-Ghalayani, 2004]**. Les verbes semi conjugables doivent être réunis, pour pouvoir les classer, ensuite il faut définir leurs paradigmes. Mais les verbes non conjugables n'ont pas de paradigmes et il faut les regrouper et les étudier en détail surtout au niveau syntaxique qui est un peu particulier.

Pour l'achèvement du remplissage de la base, nous comptons utiliser des transformations automatiques des contenus de ressources existantes comme la base lexicale non normalisée DIINAR **[Dichy et al, 2002]** (source non encore disponible dans notre laboratoire) ou la version électronique du dictionnaire « لِسَان العَرَبْ » (disponible dans une version Word).

Bien que le remplissage de notre base ne soit pas encore achevé, nous constatons déjà un problème relatif à sa taille qui influence non seulement l'espace de stockage requis mais aussi le temps d'accès. A présent, nous procédons par décomposition. Ainsi, la base est composée en réalité de deux bases, à savoir, une base pour le noyau et l'extension morphologique et une autre pour les paradigmes de flexion avec des pointeurs qui permettent de les lier. Cependant, une technique de compression utilisant une approche appropriée représentera éventuellement une solution.

Finalement, nous envisagerons étendre cette étude pour couvrir les autres extensions lexicales, notamment syntaxique et sémantique. *ArabicLDB* supporte ces futures extensions.

Bibliographie

[Abbes, 2004] Abbes R., ″La conception et la réalisation d'un concordancier électronique pour l'arabe″, Thèse de doctorat, INSA de Lyon, 2004.

[Abdelwahed, 1996] Abdelwahed A., "***بنية الفعل. قراءة في التصريف العربي***"، كلية الآداب و العلوم الإنسانيّة بصفاقس، تونس، 1996.

[Abdelwahed, 2002] Abdelwahed A., "قلب حرف العلّة ألفا"، مجلة بحوث جامعية، كلية الآداب و العلوم الإنسانيّة بصفاقس، تونس، 2002.

[Ammar et al, 1999] Ammar S., Dichy J., ″***Les verbes arabes***″, Collection Bescherelle, Edition HATIER, Paris, Octobre 1999.

[Blachère et al, 1975] Blachère R., Gaudefroy-Demombynes M., ″***Grammaire de l'arabe classique***″, Edition Maisonneuvre-Larose, Paris, 1975.

[Chaâben et al, 2004] Chaâben N., Belguith L., ″Implémentation du système MORPH2 d'analyse morphologique pour l'arabe non voyellé″, GEI'04, Monastir, Tunisie, 2004.

[Cynober, 2005] CYNOBER N. "Manipuler des données XML avec Java et JDOM", http://cynober.developper.com/tutoriel/java/xml/jdom/fichiers/jdom.pdf, 2005.

[Dichy et al, 2002] Dichy J., Braham A., Ghazali S., Hassoun M., "La base de connaissances linguistique DIINAR.1", International Symposium on The Processing of Arabic, Tunis, Universiyé de Manouba, 2002.

[El-Dahdah, 1999] El-Dahdeh A., "***معجم تصريف الأفعال العربيّة***"، مكتبة لبنان ناشرون، بيروت، لبنان، 1999.

[El-Dahdah, 1996] El-Dahdeh A., "***معجم قواعد اللّغة العربيّة*** *في جداول ولوحات*"، مكتبة لبنان ناشرون، بيروت، لبنان، 1999.

[El-Ghalayani, 2004] El-Ghalayani M., "***جامع الدروس العربيّة***"، المؤسسة الحديثة للكتاب، طرابلس،لبنان، 2004.

[Chaari et al, 2006] Chaari F., Gargouri B., Jmaiel M., "Vers une interface logicielle pour l'exploitation d'une base lexicale normalisée par les applications du TALN : cas de la morphologie de l'arabe", GEI'06, Hammamet, Tunisie, 2006.

[Francopoulo, 2003] Francopoulo G., "Proposition de normes des lexiques pour le traitement automatique du langue", INRIA/LORIA-ACTION SYNTAXE, Version-1.3 22 novembre 2003.

[Francopoulo, 2004] Francopoulo G., ″Proposition de normalisation de norme des lexiques pour le traitement automatique du langage″, INRIA/LORIA-ACTION SYNTAXE, Version-1.10 13 mai 2004.

[Francopoulo et al, 2006a] Francopoulo G., George M., ISO/TC 37/SC4 N130 Rev.9. Language resource management − Lexical markup framework (LMF) 2006.

[Francopoulo et al, 2006b] Francopoulo G., Declerck T., Monachini M., Romary L., "The relevance of standards for research infrastructures", LREC 2006, Genoa, Italy, May 2006.

[Miller et al, 1993] Miller George A., Beckwith R., Fellbaum C., Gross D. & Miller K.J., ″Introduction to WordNet : an on-line lexical database″, Journal of Lexicography 3, pp. 235- 244,1993.

[Polguère, 2000] Polguère A., ″Towards a theoretically-motivated general public dictionary of semantic derivations and collocations for French″, Proceedings of EURALEX'2000, Stuttgart, pp. 517-527, 2000.

[Romary, 2003] Romary L., "Action nationale INRIA Syntax (Décembre 2001 – Décembre 2003)", INRIA, 2003

[Romary et al, 2003] Romary L., Wright S., Farrar S., Gillam L.,ISO TC 37/SC4 N055, Language resource management - Implementing a data category registry within ISO TC37, 2003.

[Romary et al, 2004] Romary L., Salmon-Alt S., Francopoulo G., "Standards going concrete : from LMF to Morphalou", Workshop on Electronic Dictionaries, Coling2004,Geneva, Switzerland, 2004.

[Roques, 2001] Roques P., "***UML par la pratique***", Edition Eyrolles, Paris, 2001.

[Sérasset, 1994] Sérasset G., ″SUBLIM: un Système Universel de Bases Lexicales Multilingues et NADIA: sa spécialisation aux bases lexicales interlingues par acceptions″. Thèse de nouveau doctorat, Spécialité Informatique, Université Joseph Fourier Grenoble 1, 194 p, 1994.

[Salmon-Alt, 2004] Salmon-Alt S., "Morphalou : Lexique morphologique ouvert du français", http://loreley.loria.fr/morphalou/doc/morphalou_desc.html, 2004.

[Véronis, 1996] Véronis J., ″Multext : Multilingual Text Tools and Corpora″ http://www.lpl.univ-aix.fr/projects/multext/, 1996.

[Vossen, 1999] Vossen, P., ″EuroWordNet final report″ ; Deliverable D041, EuroWordNet, LE2-4003, LE4-8328, 1999.

[Zaafrani, 2001] Zaafrani R.,"Développement d'un environnement interactif d'apprentissage avec ordinateur de l'arabe langue étrangère", thèse Science de l'information et de la communication. Lyon : Université Lumière, Lyon 2, 2001.

[Zaysser, 1992] Zaysser L., ″Projet Eureka Genelex, couche morphologique″, Projet Eureka Genelex, Rapport Technique, 2 Novembre 1992.

Annexe A : Liste des nouvelles catégories de données selon le modèle ISO12620

verbFormAspect-0.0.0	
Status	Privatecandidate
Creation	[2006-07-01]
Change	[0000-00-00]
Profile	MorphoSyntax,
Broader Concept	/morphologicalFeature/
Conceptual Domain	/accomplished / /unaccomplished/ /imperative/
Definition [en]	
-- Source	MIRACL & LSCA
Definition [fr]	L'aspect est une des formes distinctives qui sont utilisées pour signaler la continuité et l'arrêt d'une action.
LS : english	
NS	
Name	verb form aspect
Status	
LS : french	
NS	
Name	aspect verbal
Status	
LS : arabic	
Conceptual Domain	/accomplished / /unaccomplished/ /apocopate/
NS	
Name	الصيغة
Status	

accomplished-0.0.0	
Status	Privatecandidate
Creation	[2006-07-01]
Change	[0000-00-00]
Profile	MorphoSyntax,
Broader Concept	/morphologicalFeature/
Definition [en]	
-- Source	MIRACL & LSCA
Definition [fr]	indique que l'action est achevée. Bien qu'il exprime le passé, il peut évoluer facilement au présent et futur.
LS : english	
NS	
Name	Accomplished
Status	
LS : french	
NS	
Name	Accompli
Status	
LS : arabic	
NS	
Name	المَاضِي
Status	

unaccomplished -0.0.0	
Status	Privatecandidate
Creation	[2006-07-01]
Change	[0000-00-00]
Profile	MorphoSyntax,
Broader Concept	/morphologicalFeature/
Definition [en]	
-- Source	MIRACL & LSCA
Definition [fr]	indique que l'action est en train de se réaliser, sans être accomplie. Il permet la modification des lettres principales du verbe. Il exprime le présent, et peut évoluer facilement au passé et au futur.
LS : english	
NS	
Name	Unaccomplished
Status	
LS : french	
NS	
Name	Inaccompli
Status	
LS : arabic	
NS	
Name	المُضَارَع
Status	

apocopate -0.0.0	
Status	Privatecandidate
Creation	[2006-07-01]
Change	[0000-00-00]
Profile	MorphoSyntax,
Broader Concept	/morphologicalFeature/
Definition [en]	
-- Source	MIRACL & LSCA
Definition [fr]	Apocopé est un mode, il est employé dans le conditionnel. Il se caractérise par l'absence de désinence et par des flexions courtes.
LS : english	
NS	
Name	apocopate
Status	
LS : french	
NS	
Name	apocopé
Status	
LS : arabic	
NS	
Name	المَجْزُوم
Status	

pluralBroker -0.0.0	
Status	Privatecandidate
Creation	[2006-07-01]
Change	[0000-00-00]
Profile	MorphoSyntax,
Broader Concept	/morphologicalFeature/
Definition [en]	
-- Source	MIRACL & LSCA
Definition [fr]	c'est le pluriel interne qui est utilisé avec les noms. C'est un pluriel qui n'a pas de désinence.
LS : english	
NS	
Name	pluralBroker
Status	
LS : french	
NS	
Name	Pluriel brisé
Status	
LS : arabic	
NS	
Name	جَمع تَكْسِيرْ
Status	

elInclusion-0.0.0	
Status	Privatecandidate
Creation	[2006-07-01]
Change	[0000-00-00]
Profile	MorphoSyntax,
Broader Concept	/morphologicalFeature/
Conceptual Domain	/no/ /yes/
Definition [en]	
-- Source	MIRACL & LSCA
Definition [fr]	C'est un trait morphologique pour les noms, qui prend la valeur *yes* si le nom peut être défini par l'article "الْ" et *no* sinon.
LS : english	
NS	
Name	el inclusion
Status	
LS : french	
NS	
Name	Inclusion de l'article el
Status	
LS : arabic	
NS	
Name	يُعَرَّفْ ب"ال"
Status	

voiceNoun -0.0.0	
Status	Privatecandidate
Creation	[2006-07-01]
Change	[0000-00-00]
Profile	MorphoSyntax,
Broader Concept	/morphologicalFeature/
Definition [en]	
-- Source	MIRACL & LSCA
Definition [fr]	Un nom d'une voix
LS : english	
NS	
Name	Voice noun
Status	
LS : french	
NS	
Name	nom d'une voix
Status	
LS : arabic	
NS	
Name	اسْم الصَوت
Status	

onceNoun -0.0.0	
Status	Privatecandidate
Creation	[2006-07-01]
Change	[0000-00-00]
Profile	MorphoSyntax,
Broader Concept	/morphologicalFeature/
Definition [en]	
-- Source	MIRACL & LSCA
Definition [fr]	Un nom d'une fois
LS : english	
NS	
Name	once noun
Status	
LS : french	
NS	
Name	nom d'une fois
Status	
LS : arabic	
NS	
Name	اسْم المرَّة
Status	

mannerNoun -0.0.0	
Status	Privatecandidate
Creation	[2006-07-01]
Change	[0000-00-00]
Profile	MorphoSyntax,
Broader Concept	/morphologicalFeature/
Definition [en]	
-- Source	MIRACL & LSCA
Definition [fr]	Un nom d'une manière
LS : english	
NS	
Name	manner noun
Status	
LS : french	
NS	
Name	nom d'une manière
Status	
LS : arabic	
NS	
Name	اسْم الهيْأة
Status	

timeNoun -0.0.0	
Status	Privatecandidate
Creation	[2006-07-01]
Change	[0000-00-00]
Profile	MorphoSyntax,
Broader Concept	/morphologicalFeature/
Definition [en]	
-- Source	MIRACL & LSCA
Definition [fr]	Un nom du temps
LS : english	
NS	
Name	time noun
Status	
LS : french	
NS	
Name	nom du temps
Status	
LS : arabic	
NS	
Name	اسْم الزَمَان
Status	

placeNoun -0.0.0	
Status	Privatecandidate
Creation	[2006-07-01]
Change	[0000-00-00]
Profile	MorphoSyntax,
Broader Concept	/morphologicalFeature/
Definition [en]	
-- Source	MIRACL & LSCA
Definition [fr]	Un nom d'un lieu
LS : english	
NS	
Name	place noun
Status	
LS : french	
NS	
Name	nom du lieu
Status	
LS : arabic	
NS	
Name	اسْم المَكَانْ
Status	

instrumentNoun -0.0.0	
Status	Privatecandidate
Creation	[2006-07-01]
Change	[0000-00-00]
Profile	MorphoSyntax,
Broader Concept	/morphologicalFeature/
Definition [en]	
-- Source	MIRACL & LSCA
Definition [fr]	Un nom d'un instrument
LS : english	
NS	
Name	instrument noun
Status	
LS : french	
NS	
Name	nom d'un instrument
Status	
LS : arabic	
NS	
Name	اسْم الآلة
Status	

elatif -0.0.0	
Status	Privatecandidate
Creation	[2006-07-01]
Change	[0000-00-00]
Profile	MorphoSyntax,
Broader Concept	/morphologicalFeature/
Definition [en]	
-- Source	MIRACL & LSCA
Definition [fr]	L'élatif est un aspect de l'adjectif qui en exprime une valeur supérieur, complète. Il fournit le comparatif et le superlatif.
LS : english	
NS	
Name	elative
Status	
LS : french	
NS	
Name	élatif
Status	
LS : arabic	
NS	
Name	اسْم التّفضِيل
Status	

affixedPersonalPronoun -0.0.0	
Status	Privatecandidate
Creation	[2006-07-01]
Change	[0000-00-00]
Profile	MorphoSyntax,
Broader Concept	/morphologicalFeature/
Definition [en]	
-- Source	MIRACL & LSCA
Definition [fr]	Pronom personnel affixé.
LS : english	
NS	
Name	affixed personal pronoun
Status	
LS : french	
NS	
Name	pronom personnel affixé
Status	
LS : arabic	
NS	
Name	ضَمير مُتَّصِلْ
Status	

conditionnelPronoun -0.0.0	
Status	Privatecandidate
Creation	[2006-07-01]
Change	[0000-00-00]
Profile	MorphoSyntax,
Broader Concept	/morphologicalFeature/
Definition [en]	
-- Source	MIRACL & LSCA
Definition [fr]	Un pronom conditionnel
LS : english	
NS	
Name	conditional pronoun
Status	
LS : french	
NS	
Name	pronom conditionnel
Status	
LS : arabic	
NS	
Name	اسْم الشّرط
Status	

allusivePronoun -0.0.0	
Status	Privatecandidate
Creation	[2006-07-01]
Change	[0000-00-00]
Profile	MorphoSyntax,
Broader Concept	/morphologicalFeature/
Definition [en]	
-- Source	MIRACL & LSCA
Definition [fr]	Un pronom allusif
LS : english	
NS	
Name	allusive pronoun
Status	
LS : french	
NS	
Name	pronom allusif
Status	
LS : arabic	
NS	
Name	اسْم الكِنَاية
Status	

masdar -0.0.0	
Status	Privatecandidate
Creation	[2006-07-01]
Change	[0000-00-00]
Profile	MorphoSyntax,
Broader Concept	/morphologicalFeature/
Definition [en]	
-- Source	MIRACL & LSCA
Definition [fr]	Masdar exprime l'idée verbale sous une forme abstraite.
LS : english	
NS	
Name	masdar
Status	
LS : french	
NS	
Name	masdar
Status	
LS : arabic	
NS	
Name	مَصْدَرْ
Status	

diminutiveNoun -0.0.0	
Status	Privatecandidate
Creation	[2006-07-01]
Change	[0000-00-00]
Profile	MorphoSyntax,
Broader Concept	/morphologicalFeature/
Definition [en]	
-- Source	MIRACL & LSCA
Definition [fr]	Un nom diminutif
LS : english	
NS	
Name	diminutive noun
Status	
LS : french	
NS	
Name	nom diminutif
Status	
LS : arabic	
NS	
Name	اسْم التّصغير
Status	

diminutiveNoun -0.0.0	
Status	Privatecandidate
Creation	[2006-07-01]
Change	[0000-00-00]
Profile	MorphoSyntax,
Broader Concept	/morphologicalFeature/
Definition [en]	
-- Source	MIRACL & LSCA
Definition [fr]	Un nom de relation
LS : english	
NS	
Name	relation noun
Status	
LS : french	
NS	
Name	nom de relation
Status	
LS : arabic	
NS	
Name	النّسبة
Status	

intensive -0.0.0	
Status	Privatecandidate
Creation	[2006-07-01]
Change	[0000-00-00]
Profile	MorphoSyntax,
Broader Concept	/morphologicalFeature/
Definition [en]	
-- Source	MIRACL & LSCA
Definition [fr]	Un nom intensif
LS : english	
NS	
Name	intensive
Status	
LS : french	
NS	
Name	intensif
Status	
LS : arabic	
NS	
Name	أمثلة المُبَالغة
Status	

coordinationParticle-0.0.0	
Status	Privatecandidate
Creation	[2006-07-01]
Change	[0000-00-00]
Profile	MorphoSyntax,
Broader Concept	/morphologicalFeature/
Definition [en]	
-- Source	MIRACL & LSCA
Definition [fr]	Une particule de coordination
LS : english	
NS	
Name	coordination particle
Status	
LS : french	
NS	
Name	particule de coordination
Status	
LS : arabic	
NS	
Name	حَرْفْ جَرّ
Status	

distinctiveParticle-0.0.0	
Status	Privatecandidate
Creation	[2006-07-01]
Change	[0000-00-00]
Profile	MorphoSyntax,
Broader Concept	/morphologicalFeature/
Definition [en]	
-- Source	MIRACL & LSCA
Definition [fr]	Une particule distinctive
LS : english	
NS	
Name	distinctive particle
Status	
LS : french	
NS	
Name	Particule distinctive
Status	
LS : arabic	
NS	
Name	حَرْفْ
Status	

relativeParticle-0.0.0	
Status	Privatecandidate
Creation	[2006-07-01]
Change	[0000-00-00]
Profile	MorphoSyntax,
Broader Concept	/morphologicalFeature/
Definition [en]	
-- Source	MIRACL & LSCA
Definition [fr]	Une particule relative
LS : english	
NS	
Name	relative particle
Status	
LS : french	
NS	
Name	particule relative
Status	
LS : arabic	
NS	
Name	حَرْفْ المَوْصُول
Status	

conditionalParticle-0.0.0	
Status	Privatecandidate
Creation	[2006-07-01]
Change	[0000-00-00]
Profile	MorphoSyntax,
Broader Concept	/morphologicalFeature/
Definition [en]	
-- Source	MIRACL & LSCA
Definition [fr]	Une particule conditionnelle
LS : english	
NS	
Name	conditional particle
Status	
LS : french	
NS	
Name	particule conditionnelle
Status	
LS : arabic	
NS	
Name	حَرْفْ شَرْط
Status	

root -0.0.0	
Status	Privatecandidate
Creation	[2006-07-01]
Change	[0000-00-00]
Profile	MorphoSyntax,
Broader Concept	/morphologicalFeature/
Definition [en]	
-- Source	MIRACL & LSCA
Definition [fr]	Une racine est purement consonantique, elle est formée par une suite de trois ou quatre (ou même cinq pour les noms) consonnes formant la base du mot.
LS : english	
NS	
Name	root
Status	
LS : french	
NS	
Name	racine
Status	
LS : arabic	
NS	
Name	الجذر
Status	

scheme -0.0.0	
Status	Privatecandidate
Creation	[2006-07-01]
Change	[0000-00-00]
Profile	MorphoSyntax,
Broader Concept	/morphologicalFeature/
Definition [en]	
-- Source	MIRACL & LSCA
Definition [fr]	Le schème est un mot composé de trois consonnes ف, ع, et ل, qui sont vocalisées et qui peuvent être augmentées par d'autres lettres (préfixe, suffixe, et infixe). Le schème joue un rôle très important dans le processus de génération des formes dérivées d'une racine.
LS : english	
NS	
Name	scheme
Status	
LS : french	
NS	
Name	schème
Status	
LS : arabic	
NS	
Name	الوَزن
Status	

Annexe B : Squelette de classification des verbes arabes

فِعْلٌ صَحِيحْ																
الوزن	سَالِمْ			مُضَاعَفْ			مُضَاعَفْ + مَهْمُوزْ			مَهْمُوزْ						
										الفَاء			العَيْنْ			اللاَّمْ
	*	+ن	+ت	*	+ن	+ت	*	ن	ت	*	+ن	+ت	*	+ن	+ ت	+ن
فُعَلَ ـُ	1	25	43	59	71	79	86			98	112					139
فُعَلَ ـِ	2	26	44	60	72	80	87	95		99	113	120	121		138	
فُعَلَ ـَ	3	27	45	61						100			122	133		140
فُعُلَ ـُ	4	28	46	62					97	101			123			141
فُعِلَ ـَ	5	29	47	63			88			102	114		124			142
فُعِلَ ـِ	6															
فَعَّلَ	7	30	48	64	73	81	89	96		103	115		125	134		143
فَاعَلَ	8	31	49	65	74					104			126	135		144
أَفْعَلَ	9	32	50	66	75	82	90			105	116		127			145
تَفَعَّلَ	11/10	33	51							106			128	10		146
تَفَاعَلَ	13/12	34	52	67	76	83				107			129	136		148/147
اِنْفَعَلَ	15/14	35	53	68		84				108						149
اِفْتَعَلَ	15/14	36/35		69/68	77		91			15/109	117		131/130	137		150/149
اِفْعَلَّ	16	37	54													
اِسْتَفْعَلَ	17	38	55	70	78	85	92			110	118		132			151
اِفْعَوْعَلَ	18	39														
اِفْعَوَّلَ	19															
اِفْعَالَّ	20	40	56													
فَعْلَلَ	21	41	57				93			111	119		**22**			153/152 157
تَفَعْلَلَ	22	42	58	**22**			94									154
اِفْعَنْلَلَ	23															155
اِفْعَلَلَّ	24															156 158

*** : الجذر الأصل بدون عوامل أخرى** **ن : اَخِرُهُ "ن"** **ت : اَخِرُهُ "ت"،**

فِعْلٌ مُعْتَلٌ																
الفاء								العَيْنْ								
و			ي		مَهْمُوزْ			و			ي			مَهْمُوزْ		
*	+ن	+ت	*	+ن	+مل	+مع		*	+ن	+ت	*	+ت	+ن	*	+مف	+مل
					و	ي	و									
1			1	25				182	184	185					202	
159	166	168	169				177							186	203	206
160				174	179							198		187		207
161			170	175												208
162			171	176	181/180									188	102	209
163	167		172			178										
7								7						7	103	143
8	31							8								144
164			173					9				199	193	9/189		210
10								10							106	146
12				34				11							204	147
14												200	194	190	205	149
15	36				150/149									14/189		211
								16								
17								17				201	195	191		212
165								183					196			
21								21					41			
22																
								23								

مع: مَهْمُوزْ العَيْنْ، مل : مَهْمُوزْ اللاَّمْ، ت : اَخِرُهُ "ت"، ن : اَخِرُهُ "ن"، و : مُعتَل بالوَاو، ي : مُعتَل بالياَء

فِعْلٌ مُعْتَلٌ								
	اللاَّمْ					لَفِيفْ مَقْرُونْ		لَفِيفْ مَقْرُوقْ
	و	ي	مَهْمُوزْ			+ مع	*	+ مع	*
			ف	ع	ل				
فَعَلَ ـُ	213		230	239					
فَعَلَ ـِ		216	231			231	216	253	249
فَعَلَ ـَ		217	232	241/240					
فَعُلَ ـُ	214								
فَعِلَ ـَ		218	233				257/256/218		250
فَعِلَ ـِ									251
فَعَّلَ		219	234	242			219		219
فَاعَلَ		220	235	243		235	220		220
أَفْعَلَ		221		245/244			258/221		221
تَفَعَّلَ		222	236	246			222		222
تَفَاعَلَ		223		247			223		223
اِنْفَعَلَ		224		248			224		
اِفْتَعَلَ		225/224	237	248				254	252/225
اِفْعَلَّ		226							
اِسْتَفْعَلَ		227	238			238	227	255	227
اِفْعَوْعَلَ	215								
اِفْعَوَّلَ							259		
فَعْلَلَ	21	228/21			152				21
تَفَعْلَلَ	22	22			153				22
اِفْعَنْلَلَ		229							
اِفْعَلَلَّ									

*** : الجذر الأصل بدون عوامل أخرى م : مَهْمُوزْ ف : الفاء ع : العَيْنْ**

Annexe C : Verbes types des paradigmes

نمُوذج	عدد	نمُوذج	عدد	نمُوذج	عدد	نمُوذج	عدد	نمُوذج	عدد	نمُوذج	عدد	نمُوذج	عدد	نمُوذج	عدد	نمُوذج	عدد
كَتَبَ	1	فَتَنَ	30	مَدَّ	59	أَمَّ	88	اِئْتَمَنَ	117	تَبَرَّأَ	146	يَمَنَ	175	تَآدَى	204	أَسِيَ	233
ضَرَبَ	2	رَاهَنَ	31	فَرَّ	60	أَمَّمَ	89	اِسْتَأْذَنَ	118	تَخَاطَأَ	147	يَقِنَ	176	اِئْآد	205	أَدَّى	234
فَتَحَ	3	أَتْقَنَ	32	عَضَّ	61	آجَّ	90	طَأْمَنَ	119	اِدَّارَأَ	148	وَأَدَ	177	جَاءَ	206	آخَى	235
كَرُمَ	4	تَحَصَّنَ	33	هَمَّ	62	اِئْتَجَّ	91	أَبَتَ	120	اِلْتَجَأَ	149	بَئِسَ	178	دَاءَ	207	تَأَخَّى	236
عَلِمَ	5	تَرَاهَنَ	34	ظَلَّ	63	اِسْتَأَبَّ	92	رَأَسَ	121	اِتَّجَأَ	150	وَبَأَ	179	هَيُؤَ	208	اِئْتَسَى	237
حَسِبَ	6	اِنْدَفَنَ	35	عَلَّمَ	64	لَأْلَأَ	93	سَأَلَ	122	اِسْتَقْرَأَ	151	وَطِئَ	180	شَاءَ	209	اِسْتَأْوَى	238
عَلَّمَ	7	اِدَّفَنَ	36	شَادَّ	65	تَكَأْكَأَ	94	رَؤُفَ	123	سَمْأَلَ	152	وَبِئَ	181	أَسَاءَ	210	شَأَى	239
صَارَعَ	8	اِحْصَنَّ	37	أَحَبَّ	66	أَنَّ	95	بَئِسَ	124	كَرْفَأَ	153	قَالَ	182	اِسْتَاءَ	211	رَأَى	240
أَكْرَمَ	9	اِسْتَبْطَنَ	38	تَشَادَّ	67	أَنَنَ	96	رَأَسَ	125	تَدَرْبَأَ	154	اِحْوَالَّ	183	اِسْتَنَاءَ	212	فَأَى	241
تَفَقَّدَ	10	اِدْجَوْجَنَ	39	اِنْشَقَّ	68	أَتَّ	97	لاءَمَ	126	اِسْلَنْطَأَ	155	كَانَ	184	دَعَا	213	رَأَى	242
اِسَّمَّعَ	11	اِشْعَانَّ	40	اِظَّنَّ	69	أَخَذَ	98	أَثْأَرَ	127	اِزْرَأَمَّ	156	فَاتَ	185	سَهُوَ	214	شَاءَى	243
تَنَازَعَ	12	بَرْهَنَ	41	اِسْتَخَفَّ	70	أَثَرَ	99	تَرَأَمَ	128	طَمْأَنَ	157	بَاعَ	186	اِشْرَوْرَى	215	أَرَى	244
اِدَّارَكَ	13	تَشَيْطَنَ	42	جَنَّ	71	أَلَهَ	100	تَفَاءَلَ	129	اِطْمَأَنَّ	158	جَادَ	187	رَمَى	216	أَفْأَى	245
اِنْطَلَقَ	14	سَكَتَ	43	رَنَّ	72	أَصُلَ	101	اِرْتَأَسَ	130	وَصَلَ	159	خَافَ	188	سَعَى	217	تَرَأَى	246
اِمَّدَحَ	15	فَلَتَ	44	جَنَّنَ	73	أَلِفَ	102	اِتَّأَمَ	131	وَضَعَ	160	أَثَارَ	189	نَسِيَ	218	تَشَاءَى	247
اِصْفَرَّ	16	بَغَتَ	45	مَانَّ	74	أَجَّلَ	103	اِسْتَرْأَفَ	132	وَسُمَ	161	اِنْصَاعَ	190	بَكَى	219	اِرْتَأَى	248
اِسْتَقْبَلَ	17	مَقَتَ	46	أَكَنَّ	75	آجَرَ	104	مَأَنَ	133	وَجِعَ	162	اِسْتَعَادَ	191	نَادَى	220	وَفَى	249
اِحْدَوْدَبَ	18	بَهِتَ	47	تَحَانَّ	76	آنَسَ	105	مَأَنَ	134	وَثِقَ	163	لاَنَ	192	أَسْمَى	221	وَنِيَ	250
اِقْلَوَّدَ	19	سَكَّتَ	48	اِقْتَنَّ	77	تَأَبَّطَ	106	مَاءَنَ	135	أَوْحَشَ	164	أَلاَنَ	193	تَرَجَّى	222	وَلِيَ	251
اِدْهَامَّ	20	بَاغَتَ	49	اِسْتَحَنَّ	78	تَأَزَرَ	107	تَمَاءَنَ	136	اِيْرَاقَّ	165	اِزْدَانَ	194	تَصَابَى	223	اِيْتَشَى	252
فَعْلَلَ	21	أَثْبَتَ	50	كَتَّ	79	اِئْأَطَرَ	108	اِشْتَأَنَ	137	وَزَنَ	166	اِسْتَعَانَ	195	اِرْتَمَى	224	وَأَى	253
تَزَحْلَقَ	22	تَزَمَّتَ	51	شَتَّ	80	اِئْتَجَرَ	109	نَأَتَ	138	وَهِنَ	167	اِزْيَانَّ	196	اِدَّعَى	225	اِتَّأَى	254
اِحْرَنْجَمَ	23	تَهَافَتَ	52	بَتَّتَ	81	اِسْتَأْجَرَ	110	هَنَأَ	139	وَقَتَ	168	بَاتَ	197	اِرْعَوَى	226	اِسْتَوْأَى	255
اِقْشَعَرَّ	24	اِنْفَلَتَ	53	أَبَتَّ	82	تَأَلَّلَ	111	بَدَأَ	140	يَتَمَ	169	بَاتَ	198	اِسْتَدْعَى	227	حَيِيَ	256
خَزَنَ	25	اِكْمَتَّ	54	تَكَاتَّ	83	أَبَنَ	112	جَرُؤَ	141	يَسُرَ	170	أَمَاتَ	199	دَهْدَى	228	حَيَّ	257
عَلَنَ	26	اِسْتَنْصَتَ	55	اِنْحَتَّ	84	أَمَنَ	113	خَطِئَ	142	يَقِظَ	171	اِنْصَاتَ	200	اِهْبَيَّخَ	229	أَحْيَا	258
رَهَنَ	27	اِكْمَاتَّ	56	اِسْتَشَتَّ	85	أَذِنَ	114	بَرَّأَ	143	يَئِسَ	172	اِسْتَمَاتَ	201	أَبَا	230	اِحْوَوَّى	259
حَسُنَ	28	كَتْكَتَ	57	أَبَّ	86	أَمَّنَ	115	قَارَأَ	144	أَيْقَظَ	173	آلَ	202	أَوَى	231		
حَزِنَ	29	تَعَفْرَتَ	58	أَتَّ	87	آمَنَ	116	أَقْرَأَ	145	يَمُنَ	174	آذَ	203	أَبَى	232		

Annexe D : Le DTD de la norme LMF

Nous allons présenter le DTD proposé dans la révision 9 de la norme LMF.

```
<?xml version='1.0' encoding="UTF-8"?>
<!-- DTD for LMFNLP packages-->

<!-- Core package-->
<!ELEMENT Database (DC*, Lexicon+, SenseAxis*, TransferAxis*, ExampleAxis*)>
<!ATTLIST Database dtdVersion CDATA #FIXED "1.0">
<!ELEMENT Lexicon (LexiconInformation, LexicalEntry+, InflectionalParadigm*, MWEPattern*, Construction*, ConstructionSet*, SemanticPredicate*, Synset*)>
<!ELEMENT LexiconInformation (DC*)>
<!ELEMENT LexicalEntry (DC*, LemmatisedForm+, Sense*, EntryRelation*, SyntacticBehavior*)>
<!ATTLIST LexicalEntry
        id ID #IMPLIED>
<!ELEMENT Sense (DC*, SenseRelation*, PredicativeRepresentation*, SenseExample*, SemanticDefinition*)>
<!ATTLIST Sense
        id ID #IMPLIED
        inherit IDREFS #IMPLIED>
<!ELEMENT EntryRelation (DC*)>
<!ATTLIST EntryRelation
        targets IDREFS #REQUIRED>
<!ELEMENT SenseRelation (DC*)>
<!ATTLIST SenseRelation
        targets IDREFS #REQUIRED>

<!-- Package for Morphology -->
<!ELEMENT LemmatisedForm (DC*, ListOfComponents?, InflectedForm*, Stem*)>
<!ATTLIST LemmatisedForm
        id ID #IMPLIED
        paradigm IDREF #IMPLIED
        pattern IDREF #IMPLIED>
<!ELEMENT ListOfComponents (DC*)>
<!ATTLIST ListOfComponents
        targets IDREFS #REQUIRED>
<!ELEMENT InflectedForm (DC*)>
<!ELEMENT Stem (DC*)>

<!-- Package for inflectional paradigms -->
<!ELEMENT InflectionalParadigm (DC*, MorphologicalFeaturesCombo*)>
<!ATTLIST InflectionalParadigm
        id ID #REQUIRED>
```

```
<!ELEMENT MorphologicalFeaturesCombo (DC*, Composer*,
InflectedFormCalculator*, MorphologicalFeature*)>
<!ELEMENT Composer (DC*, MorphologicalFeature*)>
<!ELEMENT InflectedFormCalculator (DC*, Operation*)>
<!ELEMENT Operation (DC*, OperationArgument*)>
<!ELEMENT OperationArgument (DC*)>
<!ELEMENT MorphologicalFeature EMPTY>
<!ATTLIST MorphologicalFeature
                att CDATA #REQUIRED
                val CDATA #REQUIRED>

<!-- Package for MWE patterns -->
<!ELEMENT MWEPattern (DC*, Combiner*)>
<!ELEMENT Combiner (DC*, CombinerArgument*)>
<!ELEMENT CombinerArgument (DC*, Combiner*)>

<!-- Package for Syntax -->
<!ELEMENT SyntacticBehavior (DC*)>
<!ATTLIST SyntacticBehavior
                id ID #IMPLIED
                senses IDREFS #IMPLIED
                constructions IDREFS #IMPLIED
                constructionsets IDREFS #IMPLIED>
<!ELEMENT Construction (DC*, Self?, SyntacticArgument*)>
<!ATTLIST Construction
                id ID #IMPLIED
                inherit IDREFS #IMPLIED>
<!ELEMENT Self (DC*)>
<!ELEMENT SyntacticArgument (DC*)>
<!ATTLIST SyntacticArgument
                target IDREF #IMPLIED
                semargs IDREFS #IMPLIED>
<!ELEMENT ConstructionSet (DC*)>
<!ATTLIST ConstructionSet
                id ID #IMPLIED
                constructions IDREFS #IMPLIED
                inherit IDREFS #IMPLIED>

<!-- Package for Semantics -->
<!ELEMENT PredicativeRepresentation (DC*, SemanticPredicate*)>
<!ELEMENT SemanticPredicate (DC*, SemanticArgument*, PredicateRelation*)>
<!ATTLIST SemanticPredicate
```

```
                id ID #REQUIRED>
<!ELEMENT SemanticArgument (DC*)>
<!ATTLIST SemanticArgument
                id ID #REQUIRED>
<!ELEMENT PredicateRelation (DC*)>
<!ATTLIST PredicateRelation
                targets IDREFS #IMPLIED>
<!ELEMENT SenseExample (DC*)>
<!ATTLIST SenseExample
                id ID #IMPLIED>
<!ELEMENT SemanticDefinition (DC*, Proposition*)>
<!ELEMENT Proposition (DC*)>
<!ELEMENT Synset (DC*, SemanticDefinition*, SynsetRelation*)>
<!ATTLIST Synset
                id ID #IMPLIED>
<!ELEMENT SynsetRelation (DC*)>
<!ATTLIST SynsetRelation
                targets IDREFS #IMPLIED>

<!-- Package for Multilingual notations -->
<!ELEMENT SenseAxis (DC*, SenseAxisRelation*)>
<!ATTLIST SenseAxis
                id ID #IMPLIED
                senses IDREFS #IMPLIED
                synsets IDREFS #IMPLIED>
<!ELEMENT SenseAxisRelation (DC*)>
<!ATTLIST SenseAxisRelation
                targets IDREFS #REQUIRED>
<!ELEMENT TransferAxis (DC*, TransferAxisRelation*,
                SourceTest*, TargetTest*)>
<!ATTLIST TransferAxis
                id ID #IMPLIED
                synbehaviors IDREFS #IMPLIED>
<!ELEMENT TransferAxisRelation (DC*)>
<!ATTLIST TransferAxisRelation
                targets IDREFS #REQUIRED>
<!ELEMENT SourceTest (DC*)>
<!ELEMENT TargetTest (DC*)>
<!ELEMENT ExampleAxis (DC*)>
<!ATTLIST ExampleAxis
                examples IDREFS #IMPLIED>
```

```
<!-- for datcat adornment -->
<!ELEMENT DC EMPTY>
<!-- att=constant to be taken from the DCR -->
<!-- val=free string or constant to be taken from the DCR-->
<!ATTLIST DC
                att CDATA #REQUIRED
                val CDATA #REQUIRED>
```

الخُلاصة :
يَنْدَرجُ هذا البحث ضِمن الأعمَال التي تُجْرى حاليّا لِتَثْبيت مشروع المُوَاصفات العالميّة "لمف"- "إيزو"24613 الخاصّ بتَنظيم القَوَاعِد المُعجَمِيَّة. ويهتمّ هذا البحث خَاصَّة بمدى إمكانيّة تطبيق هذا المشروع على اللّغة العَرَبِيَّة.
بُحُوثُنَا مَكَّنَتنَا مِنَ اِسْتِنْتَاج أنَّ التنظيم المقترح في "لمف" لا يُرَاعِي بشكل مناسب خَصَائص اللّغة العَرَبِيَّة إلا أنّ تَطبيقُه يبقى ممكنا عبْرَ صِيَاغة بَعْض الاِختِيارَات اللُّغَوِيّة. فقد أنْجزْنَا قَاعدة مُعجمِيّة للّغة العربيّة مُطابِقة للمُوَاصفَات المقترحة تحمل اسم "أرابيك ل دب". هذه القاعدة تشْمَلُ المُسْتَوَى الصّيغمي مَعَ نَمَاذج تَصريف الأفْعَال لأكثر مِنْ 16000 فِعل.
و قد قُمنَا بِتَحْضير الأدوات اللّازمة لشحن و اِسْتِعمال هذه القاعدة. و قُمْنَا أيْضًا بتوفير مُصرّف الأفعَال الذي يعتمدُ في عمليّة التوليد على نَمَاذج التّصريف

في نِهاية هذا العمل، نقدم للمنظّمة العالمية "إيزو" بَيَانَات حَوْلَ بَعض خَصائص اللّغة العربيّة التي لم تتطرّق إليها في الوقت الحَاضر. هذه البَيَانَات تشمل بالخصوص إضَافة المُسْتَوَى الصوتي و مُرَاجعة سِجلّ أصنَاف المَعْلومات المُدرجة وفق المواصفات العالميّة "إيزو" 12620الذي ينصّ مشروع "لمف" على استعماله.

الكلمات المفاتيح :
قاعدة معجميّة للعربية، لمف : مواصفات عالمية عدد 24613، الصّيغميّة، نماذج تَصريف الأفْعَال.

Résumé :

Le présent travail s'inscrit dans le cadre des travaux qui se font actuellement sur la validation du projet LMF (Lexical Markup Framework), future norme ISO 24613 d'élaboration des bases lexicales. Il s'intéresse en particulier à l'applicabilité de LMF pour la langue arabe.

Nos investigations ont permis de constater que LMF n'appréhende pas la langue arabe selon ses propres caractéristiques. Cependant, il peut s'appliquer au cas de cette langue moyennant certains choix de modélisation linguistique. Ainsi, une base lexicale de l'arabe conforme à LMF, baptisée ArabicLDB (Arabic Lexical DataBase), a été réalisée. Cette base couvre le niveau morphologique ainsi que les modes de flexion verbale pour plus que 16 000 verbes. Des outils nécessaires pour l'acquisition et la consultation ont été développés pour assurer la gestion de la base ArabicLDB. Aussi, un conjugueur qui se base sur un module de calcul des formes fléchies a été mis en place.

Afin de prendre en considération les caractéristiques de l'arabe, non prises en considération par LMF, des renseignements ont été fournies au sous-comité TC37-SC4 de l'ISO. Ces renseignements concernent notamment l'ajout d'une extension phonologique et la mise à jour du registre des catégories de données relatif à la norme ISO 12620, utilisé par LMF.

Mots clés :

Base lexicale arabe, LMF : norme ISO 24613, morphologie, paradigme de flexion verbale.

Abstract :

The present framework belongs to recent works carried out on the validation of LMF (Lexical Markup Framework), the future norm ISO 24613 dealing with the standardisation of lexical databases. In particular, we are interested in the application of LMF to Arabic language.

According to our investigations, we note that LMF don't consider some specific features of Arabic language. However, it can be applied according to some linguistic modelling choices. Doing so, we developed ArabicLDB, a lexical databases for Arabic language which is conform to LMF. Currently, ArabicLDB deals with morphologic level as well as the inflexion paradigms for more than 16000 verbs. Furthermore, we developed some tools to manage this lexical database. In addition, a conjugation tool of verbs was been developed using the inflexion paradigms.

In order to consider the Arabic language features, those are not taken into consideration by LMF, we provided some suggestions to the sub-committee TC37-SC4 of ISO. These suggestions concern notably the consideration of a phonologic level and the updating of the data category registry relating to the norm 12620 ISO, used by LMF.

Key words :

Arabic lexical database, LMF: ISO 24613, morphology, inflected paradigm of verbs.