

PHILSOFT - Définition des modèles

Spécifications détaillées

Ce document présente la méthode de description d'un modèle de Markov au moyen d'un fichier dit de spécifications détaillées (.DEF). L'ensemble de la description peut être fournie dans le [fichier de spécification \(.DEF\)](#). Cependant on place généralement dans un [fichier bibliothèque \(.XDF\)](#) une partie des spécifications, en particulier celles qui ne dépendent pas de l'application, ce qui permet d'alléger le contenu du fichier de spécification (.DEF). La [syntaxe formelle que doivent respecter les fichiers de spécification \(.DEF\) et la bibliothèque \(.XDF\)](#) est précisée dans le chapitre 4. Le chapitre 5 donne un [schéma du fichier de spécifications et de la bibliothèque](#) qui permet de retrouver rapidement l'enchaînement et la spécification des différentes informations.

Table des matières

1. FICHIERS DE DONNÉES	5
1.1 FORMAT DES DONNÉES	5
1.2 LIGNES SPÉCIALES	6
1.3 COMMANDE LISTING.....	7
1.4 COMMANDES DE MISE AU POINT	8
2. FICHIER DE DÉFINITION (.DEF).....	11
2.1 ORDRE DES RUBRIQUES	11
2.2 SPÉCIFICATIONS SYNTAXIQUES	12
2.2.1 Organisation des spécifications	13
2.2.2 Description des spécifications	15
2.3 SPÉCIFICATIONS LEXICALES	17
2.3.1 Organisation des spécifications	17
2.3.2 Description des spécifications	18
2.4 SPÉCIFICATIONS ACOUSTIQUES.....	21
2.4.1 Organisation des spécifications	21
2.4.2 Description des spécifications	22
3. FICHIER BIBLIOTHÈQUE (.XDF).....	25
3.1 BLOC DES INITIALISATIONS	25
3.1.1 Spécifications syntaxiques.....	26
3.1.2 Spécifications lexicales	26
3.1.3 Spécifications acoustiques	27
3.2 BLOC DES MODIFICATIONS	27
3.2.1 Spécifications syntaxiques.....	28
3.2.2 Spécifications lexicales	28
3.2.3 Spécifications acoustiques	29
4. SYNTAXE DES SPÉCIFICATIONS	31
4.1 CONTENU DES FICHIERS.....	31
4.2 SPÉCIFICATIONS SYNTAXIQUES	32
4.3 SPÉCIFICATIONS LEXICALES	35
4.4 SPÉCIFICATIONS ACOUSTIQUES.....	38
4.5 ÉLÉMENTS DE BASE	41
5. SCHÉMA DES SPÉCIFICATIONS	43
5.1 FICHIER PRINCIPAL DE DÉFINITION (.DEF)	43
5.2 FICHIER BIBLIOTHÈQUE ÉVENTUEL (.XDF)	45
6. REMARQUES ET LIMITATIONS	47

1. Fichiers de données

La description du modèle de Markov à créer se fait sous forme de données ASCII, donc dans un fichier texte (utilisation d'un éditeur de texte quelconque). Ce fichier, pour être interprété correctement, doit respecter certaines normes. Les rubriques possibles, leur syntaxe et leur rôle, seront décrites dans les chapitres suivants.

Dans ce chapitre, on se limite aux informations générales, en commençant par le format d'introduction des données, qui précise les mots clés et les unités reconnus par le compilateur. Ensuite on évoque le traitement des lignes spéciales et la commande `#Listing` permettant de connaître l'état des données. Enfin, on décrit le bloc de commandes qui a servi à la mise au point du programme, et qui peut être utilisé pour récupérer certaines informations.

1.1 Format des données

Les spécifications sont regroupées par blocs de données. Chaque bloc, identifié par un ou plusieurs mots clés, débute par une accolade ouvrante, se termine par une accolade fermante, et contient un ensemble de spécifications concernant une même rubrique. Ces spécifications peuvent être soit des listes d'éléments, soit des sous-blocs, soit des informations de base.

Les listes d'éléments sont identifiées par un ou plusieurs mots clés. Ensuite, on trouve le signe "=" (égal), puis la liste des éléments et enfin un ";" (point virgule). La liste elle-même, débute par une "(" (parenthèse ouvrante), se termine par une ")" (parenthèse fermante), et contient les noms des éléments séparés par des "," (virgules). Dans le cas où la liste serait réduite à un seul élément les parenthèses sont facultatives.

Quant aux informations de base, ce sont généralement des affectations qui font intervenir le signe "=" (égal) et qui se terminent par un ";" (point-virgule). Le signe "=" (égal) signifie que l'on remplace la ou les occurrences de la partie gauche par l'expression définie dans la partie droite.

Un autre type d'objet qui revient fréquemment dans les spécifications est la notion de contexte. Ces contextes, interviennent aux niveaux lexical et acoustique et correspondent à des ensembles d'unités lexicales (généralement des phonèmes). Les contextes sont généralement délimités par des crochets "[" et "]" et peuvent être spécifiés, soit à partir des unités lexicales, soit à partir d'autres contextes déjà définis.

Les termes de base des descriptions sont en fait des chaînes de caractères qui peuvent être de deux types : soit des suites de caractères alphanumériques (`a ... z, A ... Z, 0 ... 9`) pouvant contenir les caractères "\$" (dollar) et "_" (souligné), soit des suites de caractères quelconques entre doubles quotes. Dans le premier cas, la chaîne de caractères est délimitée par des espaces ou des séparateurs tels que les caractères de ponctuation. Dans le second cas, la chaîne est délimitée par les doubles quotes, tous les caractères sont acceptés (y compris les espaces et les caractères accentués) et l'anti-slash sert de caractère d'échappement pour introduire une double quote (`\"`) ou un anti-slash (`\\`), les autres séquences étant interdites pour l'instant.

En dehors des chaînes de caractères entre quotes, les caractères d'espacement (blancs, tabulations, etc) sont éliminés. Ils servent simplement de séparateurs entre les éléments de la description. Le point d'exclamation, quant à lui, permet d'introduire un commentaire : la fin de la ligne après le point d'exclamation est ignorée.

Pour compléter ces généralités, signalons que toutes les lignes débutant par le caractère # sont considérées comme des lignes spéciales, les deux plus courantes permettant l'inclusion de fichiers et la définition de synonymes.

1.2 Lignes spéciales

Les lignes spéciales sont des lignes du fichier dont le premier caractère non blanc est un #. Ces lignes sont traitées séparément et permettent de définir des synonymes, d'inclure des fichiers, ou bien d'effectuer quelques traitements particuliers. Les commandes **#include**, **#define** et **#undef** ont un fonctionnement similaire à leurs homologues en C. Toutefois la définition de synonymes ne permet pas la définition de macros (pas d'arguments) et doit tenir sur une seule ligne.

```
#include "nom_de_fichier"
```

Cette commande permet d'inclure, à l'endroit courant, le contenu du fichier indiqué. Les champs non spécifiés dans le nom du fichier sont pris (par défaut) dans le nom du fichier courant. Après la lecture du fichier spécifié, la lecture continue sur le fichier courant. Attention : les doubles quotes, avant et après le nom du fichier, sont obligatoires.

```
#define token chaine_de_replacement
```

Cette commande permet de définir des synonymes pour éviter d'avoir à mettre des chaînes entre quotes, ou bien pour des séquences qui reviennent fréquemment. Lorsque l'élément `token` sera rencontré au cours du décodage du fichier, on effectuera (à ce moment là) le décodage de la fin de la ligne associée (`chaine_de_replacement`), puis on poursuivra le décodage normal.

```
#undef token
```

Cette commande supprime le symbole `token` de la liste des synonymes.

```
#Exit
```

Cette commande interrompt la compilation en cours.

```
#Listing [-option] [-option] [...] [mot_cle [mot_cle [pattern]]]
```

Cette commande imprime tout ou partie des informations décrivant l'état de la compilation : réseaux, listes, etc. L'ensemble des informations pouvant nécessiter beaucoup de pages, il est possible de sélectionner les informations à lister au moyen des mots clés et du pattern (voir la description de la commande `listing`).

```
#Purge [Donnees]
```

Cette commande permet d'effectuer le contrôle des données du niveau précisé (syntaxique, lexical ou acoustique). Elle n'est valide que dans le bloc de mise au point (voir les commandes de mise au point).

```
#Compilation [Donnees]
```

Cette commande permet d'effectuer la compilation des données du niveau précisé (syntaxique, lexical ou acoustique). Il y a un enchaînement automatique des actions requises avant la compilation demandée. Cette commande n'est valide que dans le bloc de mise au point (voir les commandes de mise au point).

1.3 Commande listing

Les commandes listing, au niveau des lignes spéciales, sont de la forme :

```
#Listing [-Option] [-Option] [...] [Premier_mot_clé [Deuxième_mot_clé [Pattern]]]
```

Les options servent uniquement à modifier les unités de sortie. Par défaut, les informations demandées sont écrites dans le fichier LOG mais pas affichées sur l'écran. L'option **-Log** valide l'écriture sur le fichier tandis que **-NoLog** la supprime. Les options **-Screen** ou **-Window** entraînent l'affichage sur l'écran (ou dans les fenêtres suivant le mode), tandis que les options **-NoScreen** ou **-NoWindow** suppriment cet affichage. Par défaut on a donc :
#Listing -Log -NoWindow

Le tableau ci-dessous résume les valeurs possibles pour les deux mots clés et donne les informations qui seront alors affichées. De plus, les premières colonnes indiquent si le listing est pertinent au niveau de la syntaxe, du lexique, de l'acoustique ou en fin de programme (cas de l'option **-Informations** sur la ligne de commande). Les mots clés peuvent être abrégés, et écrits en majuscules ou en minuscules. En cas d'abréviation, on liste toutes les informations dont le début des mots clés correspond aux abréviations. Si on ne donne aucun mot clé, on liste toutes les informations disponibles, ce qui peut prendre beaucoup de pages. L'utilisation d'un pattern permet de limiter le listing aux objets dont le nom correspond au pattern. En ce qui concerne les patterns, d'une part, les majuscules et les minuscules ne sont pas équivalentes, et d'autre part, il peut y avoir des problèmes avec le traitement des espaces ! Les jokers "%", "?" et "*" remplacent respectivement un ("% " ou "? ") ou plusieurs ("*") caractères quelconques.

S y n t	L e x i c	A c o u	I n f	Commandes possibles (Mots clés)	Description des informations fournies
#	#	#	#	Sommaire Memoire	Liste la taille mémoire occupée par les réseaux, les symboles et les ensembles.
#	#	#	#	Status Symboles	Liste l'état des listes d'éléments ou d'unités : déclarées, définies ou utilisées.
#			#	Entrees Syntaxiques	Liste les noms des entrées syntaxiques (points d'entrée du réseau principal) définies.
#			#	Application	Liste le réseau principal. Le contenu dépend de l'état de la compilation.
			#	Regroupements	Liste les regroupements des états du réseau principal au niveau acoustique.
#				Elements Syntaxiques	Liste les éléments syntaxiques (entrées syntaxiques, non terminaux et terminaux).
#				Reseaux Syntaxiques	Liste les réseaux syntaxiques associés aux entrées syntaxiques et aux éléments non terminaux.

#		#	Mots Syntaxiques	Liste les mots syntaxiques de l'application.
#		#	Unites Syntaxiques	Liste les unités syntaxiques de la modélisation.
	#	#	Unites Lexicales	Liste les unités lexicales de la modélisation.
	#		Contextes Lexicaux	Liste les contextes définis ou utilisés au niveau lexical.
	#		Reseaux Lexicaux	Liste les réseaux lexicaux associés aux unités syntaxiques.
		#	Noms Modeles	Liste les noms des modèles acoustiques définis.
		#	Etats Modeles	Liste les noms des états de l'ensemble des modèles acoustiques.
		#	Fonctions Modeles	Liste les noms des fonctions de l'ensemble des modèles acoustiques.
		#	Contextes Acoustiques	Liste les contextes définis ou utilisés au niveau acoustique.
		#	Modeles Acoustiques	Liste les réseaux correspondant aux modèles acoustiques.
		#	Reseaux Acoustiques	Liste les réseaux acoustiques associés aux unités lexicales.
		#	Noms Etats	Liste les noms des états des réseaux acoustiques.
		#	Noms Fonctions	Liste les noms des fonctions associées aux réseaux acoustiques.

1.4 Commandes de mise au point

Un bloc spécial de spécifications à servi pour la mise au point du logiciel de compilation. Ce bloc est identifié par le mot clé **Commandes** et ne peut contenir que des lignes spéciales (commençant par le caractère "#"). Il est facultatif, et, s'il est utilisé, il doit se trouver impérativement en fin de fichier (fichier principal ou bibliothèque). Son seul et unique but est de permettre l'introduction de commandes spéciales afin de connaître l'état des données à différentes étapes de la compilation. En fait, les traitements effectués après le décodage des fichiers, c'est-à-dire le contrôle des informations et la compilation du modèle, reviennent à exécuter la séquence suivante de commandes spéciales :


```
Commandes {  
    #Purge Syntaxe  
    #Purge Lexique  
    #Purge Acoustique  
    #Compilation Syntaxe  
    #Compilation Lexique  
    #Compilation Acoustique  
}
```

L'intérêt de ce bloc est de pouvoir alterner ces commandes spéciales avec des commandes listing afin de suivre l'évolution des données à chaque étape du traitement. L'appel des commandes spéciales entraîne l'enchaînement automatique des traitements nécessaires à son exécution correcte (remarque : des marqueurs internes au programme permettent de savoir si chacune des étapes a déjà été effectuée ou non).

2. Fichier de définition (.DEF)

Ce fichier, obligatoire, contient toutes les informations spécifiques de l'application. Les informations générales, ou communes à plusieurs applications, peuvent être fournies dans un fichier bibliothèque, et n'ont donc pas, dans ce cas, à être re-spécifiées pour chacune des applications.

Le programme de compilation décode les informations fournies dans les fichiers (fichier principal et fichier bibliothèque) dans l'ordre suivant :

- bloc des "Initialisations" (ou bloc "Avant") du fichier bibliothèque,
- le fichier principal (blocs **syntaxique**, **lexical** et **acoustique**),
- et enfin le bloc des "Modifications" (ou bloc "Après") du fichier bibliothèque.

Dans la suite de ce chapitre, nous décrirons les informations fournies dans le fichier principal et qui sont regroupées en trois niveaux de description: syntaxique, lexical et acoustique. Le niveau syntaxique précise les phrases valides (ou correctes) de l'application. Le niveau lexical décrit chaque unité syntaxique en termes d'unités lexicales (généralement des phonèmes). Enfin, le niveau acoustique précise la modélisation acoustique des unités lexicales. Dans le cas où le niveau lexical est omis, les unités lexicales implicites sont identiques aux unités syntaxiques.

Nous allons donc décrire successivement ces trois niveaux en commençant par quelques généralités sur l'ordre des rubriques. Pour chaque niveau, nous décrirons l'ordre "standard" des rubriques et nous préciserons, pour le niveau syntaxique, les principales combinaisons possibles. Nous détaillerons également le rôle de chacune des rubriques.

2.1 Ordre des rubriques

Il est en général possible de combiner les rubriques d'un niveau donné dans un ordre différent de l'ordre présenté, ou d'employer plusieurs fois une même rubrique. Pour savoir si un ordre de présentation est acceptable ou non, il est nécessaire de connaître le rôle (ou tout au moins le type) de chacune des rubriques. On distingue quatre types de rubriques qui correspondent respectivement à :

- des **déclarations** qui consistent à déclarer les unités (ou éléments) de chaque type et à les insérer dans les listes correspondantes,
- des **descriptions** qui associent des informations aux unités d'un type donné (couple syntaxique, expression syntaxique, expression lexicale, etc),
- des blocs qui combinent à la fois les **déclarations** et les **descriptions** des unités d'un type donné,
- des **modifications** qui sont appliquées aux descriptions fournies préalablement. Ces rubriques doivent être obligatoirement les dernières du niveau correspondant.

L'ordre normal des rubriques consiste donc à déclarer les unités, puis à fournir les descriptions de ces unités, et enfin, à effectuer les modifications indispensables. On peut évidemment avoir

des déclarations après des descriptions, à condition qu'elles correspondent à de nouvelles unités.

L'utilisation des rubriques de déclaration est généralement facultative. La seule obligation concerne les unités lexicales lorsque l'on utilise des règles ou des modélisations contextuelles. Lorsque l'on a déclaré une ou plusieurs unités (ou éléments) d'une liste, il faut alors déclarer toutes les autres unités du même type; c'est-à-dire que l'on doit, soit déclarer toutes les unités d'une liste, soit n'en déclarer aucune. Enfin, toutes les unités déclarées dans le fichier de définition doivent être utilisées.

Compte tenu de ces quelques remarques, les principaux "warnings" sont :

- la déclaration d'une unité déjà déclarée (double déclaration),
- la déclaration d'une unité après l'avoir utilisée (ordre des blocs),
- l'utilisation ou la description d'une unité non déclarée alors que l'on a déclaré d'autres unités du même type,
- la non utilisation d'unités déclarées.

Et les principales erreurs sont (en dehors des séparateurs) :

- une incompatibilité de type (éléments syntaxiques et états des modèles acoustiques),
- des déclarations ou descriptions après des modifications (ordre des blocs).

2.2 Spécifications syntaxiques

Le bloc syntaxique permet de spécifier les phrases "valides", c'est-à-dire celles que le système de reconnaissance pourra reconnaître (et que l'application traitera). Pour des raisons pratiques on distingue les unités syntaxiques qui servent à la modélisation et les mots syntaxiques qui sont en fait des chaînes de caractères qui seront affichées lors de la reconnaissance. En conséquence, le bloc syntaxique traite des éléments syntaxiques qui sont des couples *< mot_syntaxique , unité_syntaxique >*. A l'issue du bloc syntaxique, les mots syntaxiques ne seront plus modifiés. Par contre, les unités syntaxiques seront soit décrites au niveau lexical, soit directement modélisées au niveau acoustique.

Généralement les mots et les unités syntaxiques sont identiques. Par exemple la reconnaissance de l'unité syntaxique "Oui" entraînera l'affichage du mot syntaxique "Oui". Cependant, il est très utile d'avoir des unités syntaxiques qui n'entraînent aucun affichage, le cas le plus courant concernant les pauses. Une autre exception concerne les multi-modèles où l'on emploie plusieurs unités syntaxiques différentes pour modéliser un même mot syntaxique.

Un autre point à mentionner à propos du niveau syntaxique concerne la définition de plusieurs points d'entrée pour le réseau principal. Ces différents points d'entrée permettent de restreindre les "phrases" valides à un instant donné (en phase de reconnaissance) sans avoir à modifier le réseau complet de l'application.

2.2.1 Organisation des spécifications

Plusieurs formes de spécification sont possibles. Les 2 premiers modes sont disponibles essentiellement pour des raisons de compatibilités avec des versions antérieurs du compilateur de modèles. Ces formes sont à éviter dans les nouvelles définitions de modèles. Seule la 3^{ième} forme de spécification est préconisée.

La forme de spécification de la syntaxe qui entraîne le moins de contrôles consiste à ne pas déclarer le type des objets manipulés. Dans ce cas, on définit uniquement l'ensemble des **productions syntaxiques** et on applique, si nécessaire, un certain nombre de **règles de substitution** portant soit sur les **mots syntaxiques**, soit sur les **unités syntaxiques**.

<pre> Syntaxe { Productions { Elément_non_terminal = Expression_syntaxique ; Entrée_syntaxique = Expression_syntaxique ; } Substitutions Mots { Ancien_mot_syntaxique = Nouveau_mot_syntaxique ; } Substitutions Unites { Ancienne_unité_syntaxique = Nouvelle_unité_syntaxique ; } } </pre>	<p>Description des productions syntaxiques.</p> <p>Modifications portant sur les mots syntaxiques.</p> <p>Modifications portant sur les unités syntaxiques.</p>
---	---

Dans cette approche, le type des **éléments syntaxiques** est déterminé en examinant la position relative des termes par rapport au signe "=" (égal). Les termes qui apparaissent uniquement dans les parties gauches des productions sont considérés comme des **points d'entrée** du réseau. Ceux qui apparaissent à droite et à gauche (dans des expressions différentes) constituent les éléments **non terminaux**. Quant à ceux qui n'apparaissent qu'à droite des productions, ils constituent les **éléments terminaux** de la syntaxe. Ils sont alors remplacés par les couples `< mot_syntaxique , unité_syntaxique >` dont les composantes sont identiques au nom de l'**élément terminal** concerné.

On peut forcer le contrôle des éléments utilisés en les déclarant au préalable. Les déclarations se font indépendamment pour chacun des types concernés : éléments terminaux, éléments non terminaux et entrées syntaxiques. L'enchaînement des rubriques est alors le suivant :

<pre> Syntaxe { Elements Terminaux = (Elément_terminal , ...) ; Elements Non_Terminaux = (Elément_non_terminal , ...) ; Entrees Syntaxiques = (Entrée_syntaxique , ...) ; Productions { Elément_non_terminal = Expression_syntaxique ; Entrée_syntaxique = Expression_syntaxique ; } Substitutions Mots { Ancien_mot_syntaxique = Nouveau_mot_syntaxique ; } Substitutions Unites { Ancienne_unité_syntaxique = Nouvelle_unité_syntaxique ; } } </pre>	<p>Déclaration des éléments syntaxiques.</p> <p>Déclaration des sous-syntaxes.</p> <p>Déclaration des entrées du réseau principal.</p> <p>Description des productions syntaxiques.</p> <p>Modifications portant sur les mots syntaxiques.</p> <p>Modifications portant sur les unités syntaxiques.</p>
---	--

Dans ce cas, un contrôle est exercé sur les éléments d'un type donné, si et seulement si les mots clés identifiant ce type sont utilisés. Le contrôle sur les éléments de ce type est alors total, c'est-à-dire qu'un "warning" est généré pour tout élément de ce type non explicitement déclaré ou non utilisé. Si, par exemple, seule la rubrique **Entrees Syntaxiques** apparaît dans le bloc syntaxique, le programme vérifiera que toutes les entrées syntaxiques sont effectivement déclarées et utilisées, par contre aucun contrôle ne sera effectué sur les éléments terminaux et non terminaux.

Une autre forme de spécification de la syntaxe consiste à utiliser les rubriques combinant déclaration et description. **On déclare et on définit** d'abord tous les éléments syntaxiques qui sont en fait les **éléments terminaux** de la syntaxe, puis les **éléments non terminaux** qui correspondent à des sous-syntaxes, et enfin les **entrées syntaxiques** (i.e. les points d'entrée du réseau principal).

<pre> Syntaxe { Elements Terminaux { Elément_terminal = < Mot_syntaxique , Unité_syntaxique > ; } Elements Non_Terminaux { Elément_non_terminal = Expression_syntaxique ; } Entrees Syntaxiques { Entrée_syntaxique = Expression_syntaxique ; } } </pre>	<p>Déclaration et description des éléments syntaxiques.</p> <p>Déclaration et description des sous-syntaxes.</p> <p>Déclaration et description des points d'entrée du réseau principal.</p>
---	---

2.2.2 Description des spécifications

Nous allons maintenant décrire les différentes rubriques du niveau syntaxique en commençant par les déclarations. Ces déclarations se font séparément pour les **éléments terminaux**, les **éléments non terminaux** et les **entrées syntaxiques**. Toutes ces déclarations (sans description associée) se font sous forme de listes d'éléments. Les éléments de la liste sont séparés par des "," (virgules), et la liste complète est mise entre parenthèses "(" et ")". Toutefois, lorsque la liste est réduite à un seul élément les parenthèses sont facultatives.

Elements Terminaux = (Elément_terminal , ...) ;	Déclaration des éléments syntaxiques.
Elements Non_Terminaux = (Elément_non_terminal , ...) ;	Déclaration des sous-syntaxes.
Entrées Syntaxiques = (Entrée_syntaxique , ...) ;	Déclaration des points d'entrée du réseau principal.

Les éléments déclarés sont insérés dans la liste correspondant à leur type, et on vérifie qu'il n'y a pas double déclaration de type. En effet, un même élément ne peut pas être à la fois, par exemple, un élément terminal et un élément non terminal. On contrôle également que les chaînes de caractères identifiant les éléments ne sont pas vides. Attention, il ne faut pas oublier de mettre les chaînes de caractères entre double-quotes si elles comportent des caractères particuliers ou accentués.

La description des productions syntaxiques, lorsque les éléments sont déjà déclarés ou bien lorsque l'on ne désire pas contrôler les types, s'effectue au moyen du bloc "**Productions**". Elle consiste à décrire les **éléments non terminaux** et les **entrées syntaxiques** par des expressions syntaxiques qui font intervenir des **éléments terminaux** et éventuellement des **éléments non terminaux** (mais pas d'entrées syntaxiques). La description est exprimée par le signe "=" (égal) et chaque règle de production doit être terminée par un ";" (point-virgule).

Productions { Elément_non_terminal = Expression_syntaxique ; Entrée_syntaxique = Expression_syntaxique ; }	Description des productions syntaxiques.
--	--

Les **expressions syntaxiques** permettent de définir des successions d'éléments syntaxiques ainsi que des alternatives. Les bouclages sont prévus au niveau de la compilation mais la récursivité (même cachée) est interdite puisque l'on désire obtenir un seul réseau syntaxique en fin de compilation.

Expression_syntaxique	:=	Séquence_syntaxique Séquence_syntaxique + Expression_syntaxique	Séquence simple ou avec des alternatives.
Séquence_syntaxique	:=	Boucle_syntaxique Boucle_syntaxique . Séquence_syntaxique	Élément simple ou suite d'éléments.
Boucle_syntaxique	:=	Elément_syntaxique Elément_syntaxique *	Sans bouclage. Avec bouclage.

Elément_syntaxique	:=	(Expression_syntaxique) () Elément_non_terminal Elément_terminal Couple_syntaxique	Différents éléments de base reconnus et acceptés.
Couple_syntaxique	:=	< Mot_syntaxique , Unité_syntaxique >	Composantes du couple syntaxique.

Une **alternative** est exprimée par le signe "+" (plus) tandis qu'une **succession** d'éléments est indiquée par un "." (point). Un **bouclage** est exprimé par une "*" (étoile) et concerne l'élément syntaxique qui précède. L'**élément syntaxique** de base est soit une autre expression syntaxique entre parenthèses ("(" et ")"), soit une expression vide (notée "()"), soit un élément non terminal, soit un élément terminal ou bien directement un couple syntaxique. Dans ce dernier cas on précise entre crochets ("<" et ">") ses composantes : *Mot syntaxique* et *Unité syntaxique*.

Dans ce mode de description, l'ordre classique de priorité des opérateurs est respecté : un bouclage est plus prioritaire qu'une séquence d'éléments, elle même plus prioritaire qu'une alternative. Par exemple, l'expression (a . b + c . d) définit une alternative entre deux séquences qui sont a suivi de b d'une part, et c suivi de d d'autre part. L'utilisation des parenthèses, entre lesquelles on peut mettre n'importe quelle expression, permet de modifier l'ordre d'interprétation. C'est ainsi que l'expression (a . (b + c) . d) définit la séquence a suivi d'une alternative (entre b ou c), elle même suivie de d.

Les blocs suivants combinent à la fois la déclaration des éléments et leur description. La description des **éléments non terminaux** et des **entrées syntaxiques** est identique aux descriptions fournies dans le bloc de productions. Par contre, les éléments décrits sont en même temps explicitement déclarés.

Elements Non_Terminaux { Elément_non_terminal = Expression_syntaxique ; }	Déclaration et description des sous-syntaxes.
Entrees Syntaxiques { Entrée_syntaxique = Expression_syntaxique ; }	Déclaration et description des entrées du réseau principal.

Le bloc des **éléments terminaux**, quant à lui, déclare et décrit les éléments de base de la description syntaxique. Pour chaque élément on peut préciser ses composantes : *Mot syntaxique* et *Unité syntaxique*. Quand une des composantes est identique au nom de l'élément terminal on peut la remplacer par une "*" (étoile : il s'agit du séparateur étoile et non de la chaîne de caractères "**", c'est-à-dire qu'il ne faut pas mettre les doubles quotes). Quand les deux composantes sont identiques au nom de l'élément terminal on peut soit remplacer chacune d'elles par une étoile, soit les omettre.

Elements Terminaux { Elément_terminal = < Mot_syntaxique , Unité_syntaxique > ; Elément_terminal = < Mot_syntaxique , * > ; Elément_terminal = < * , Unité_syntaxique > ; Elément_terminal = < * , * > ; Elément_terminal ; }	Déclaration et description des éléments terminaux de la syntaxe.
---	--

Les deux dernières rubriques concernent les **modifications** (plus précisément des **substitutions**) à apporter au réseau syntaxique. Ces modifications portent respectivement sur les **mots syntaxiques** (chaînes de caractères affichées lors de la reconnaissance) et sur les **unités syntaxiques** (unités à modéliser). On peut avoir autant de blocs de chaque type que l'on veut mais il doivent être les derniers du niveau syntaxique.

<pre>Substitutions Mots { Ancien_mot_syntaxique = Nouveau_mot_syntaxique ; }</pre>	Modifications portant sur les mots syntaxiques.
<pre>Substitutions Unites { Ancienne_unité_syntaxique = Nouvelle_unité_syntaxique ; }</pre>	Modifications portant sur les unités syntaxiques.

Ces règles indiquent les mots ou unités syntaxiques concernés et les chaînes de remplacement. Ce sont **toutes les occurrences** du mot ou de l'unité syntaxique qui seront modifiées. Une chaîne vide (notée "") en tant que nouveau mot syntaxique supprime ce mot, et donc n'entraîne aucun affichage lors de la reconnaissance (utile dans le cas des pauses par exemple).

2.3 Spécifications lexicales

Le bloc lexical sert à décrire chaque unité syntaxique en termes d'unités lexicales qui correspondent généralement aux phonèmes. L'utilisation de ce niveau permet d'introduire les connaissances a priori du niveau phonétique : description des mots en termes d'unités arbitraires (syllabes, phonèmes, etc) et application de règles phonologiques.

2.3.1 Organisation des spécifications

Contrairement à la syntaxe, il n'y a pas de variantes des spécifications au niveau lexical. On déclare éventuellement les unités lexicales, puis on décrit les unités syntaxiques. On définit les contextes dont on a besoin et on applique les règles de modifications contextuelles.

<pre> Lexique { Unites = (Unité_lexicale , ...) ; Contextes { Nom_contexte = (Unité_lexicale , ...) ; Nom_contexte = [Ensemble_unités] ; } Descriptions { Unité_syntaxique = Expression_lexicale ; } Alternatives { Séquence_en_contexte = Expression_lexicale ; } Substitutions { Séquence_en_contexte = Expression_lexicale ; } } </pre>	<p>Déclaration des unités lexicales.</p> <p>Déclaration et description des contextes lexicaux.</p> <p>Description des unités syntaxiques.</p> <p>Alternatives pour des séquences en contexte.</p> <p>Remplacement de séquences en contexte.</p>
--	---

La déclaration des **unités lexicales** entraîne un contrôle sur les descriptions lexicales des unités syntaxiques (qui correspondent aux mots du vocabulaire). Ces descriptions lexicales font intervenir des expressions similaires aux expressions syntaxiques.

La déclaration des unités lexicales est facultative lorsque l'on n'utilise pas de contextes. Par contre, dès que l'on définit des contextes ou que l'on applique des règles contextuelles il faut déclarer l'ensemble des unités lexicales, et décrire toutes les unités syntaxiques (y compris les silences début et fin) par des expressions lexicales.

L'application des règles contextuelles permet de modifier les descriptions lexicales. Ces règles consistent à remplacer une séquence par une expression (**substitution**) ou bien à ajouter l'expression en parallèle (**alternative**). Les règles peuvent être contextuelles, l'application de celles-ci dépendant alors du contexte.

Comme pour la syntaxe, les déclarations éventuelles (des unités lexicales) doivent précéder les descriptions, et les alternatives et les substitutions doivent se trouver en fin de bloc.

2.3.2 Description des spécifications

La première rubrique du niveau lexical concerne la déclaration des **unités lexicales** employées. Celles-ci sont déclarées sous la forme d'une liste d'unités : ensemble de termes séparés par des virgules et mis entre parenthèses. Si le mot clé **unites** est présent plusieurs fois, les termes (correspondant à chaque occurrence) sont ajoutés à la liste courante des unités lexicales.

<pre> Unites = (Unité_lexicale , ...) ; </pre>	<p>Déclaration des unités lexicales.</p>
--	--

Ici encore, les chaînes de caractères affectées aux noms des unités lexicales ne doivent pas être vides et doivent être mises entre double-quotes si elles comportent des caractères accentués ou non alphanumériques (autres que "\$" et "_" (dollar et souligné)).

Après ce bloc apparaît naturellement le bloc de spécification des **contextes**. Celui-ci permet de définir des contextes soit à partir de listes d'unités lexicales, soit à partir d'ensembles d'unités. Si la définition est faite **sous forme de listes**, les différentes unités sont séparées par des "," (virgules) et le tout est mis entre parenthèses ("(" et ")"). Par contre, si la définition est faite **sous forme d'ensembles**, on peut utiliser (ajouter ou retrancher) des contextes (ensembles d'unités) déjà définis. La description est alors mise entre crochets ("[" et "]") et on peut utiliser une "*" (séparateur étoile) pour désigner l'ensemble des toutes les unités lexicales (déjà déclarées explicitement, ou implicitement du fait de leur utilisation). La définition d'un contexte ne peut évidemment pas être récursive, c'est-à-dire que l'on ne peut pas faire intervenir le contexte en cours de définition pour préciser l'ensemble des unités.

<pre>Contextes { Nom_contexte = (Unité_lexicale , ...) ; Nom_contexte = [Ensemble_unités] ; }</pre>	Déclaration et description des contextes lexicaux.
---	--

Lorsque l'on utilise la formulation sous forme d'ensembles d'unités lexicales on considère que l'on dispose d'un certain nombre d'opérations modifiant l'ensemble courant. Les opérations sont identiques à des opérations sur des ensembles; elles consistent à ajouter ou soustraire un ensemble d'éléments (cas des contextes lexicaux définis) ou bien à ajouter ou soustraire un élément (cas des unités lexicales).

Lorsque l'on rencontre le crochet ouvrant ("[") on initialise un nouvel ensemble (il est alors vide), et le premier élément rencontré sera automatiquement ajouté. Si ce premier élément est le séparateur étoile, on ajoute l'ensemble des unités lexicales (toute la liste). Sinon (ce n'est pas une étoile) on examine si l'élément correspond à un nom de contexte, et si c'est le cas, on ajoute le contexte correspondant. Dans le cas contraire (ce n'est pas un nom de contexte), l'élément représente une unité lexicale (qui sera au besoin ajoutée à la liste). Pour les éléments suivants, le traitement dépendra du signe qui les précède, et on retranchera (cas du signe "-" (moins)) ou on ajoutera (cas du signe "+" (plus)) le contexte (ensemble d'unités) ou l'unité lexicale (si l'élément ne correspond pas à un contexte).

Contexte_lexical	:=	[Ensemble_unités]	Description d'un contexte lexical.
Ensemble_unités	:=	<pre>* Nom_contexte Unité_lexicale Ensemble_unités + Nom_contexte Ensemble_unités - Nom_contexte Ensemble_unités + Unité_lexicale Ensemble_unités - Unité_lexicale</pre>	Spécification de l'ensemble des unités lexicales définissant le contexte.

Le bloc principal des spécifications lexicales concerne la description de chacune des unités syntaxiques en termes d'unités lexicales au moyen d'expressions lexicales.

<pre>Descriptions { Unité_syntaxique = Expression_lexicale ; }</pre>	Description des unités syntaxiques.
--	-------------------------------------

Les **expressions lexicales** se présentent un peu comme les expressions syntaxiques, mais les termes de la description sont nécessairement des unités lexicales. L'ordre de priorité des opérateurs est le même que pour la syntaxe, cependant **les bouclages ne sont pas autorisés**. Toutefois on ne dispose pas de l'équivalent des sous syntaxes (éléments non terminaux), ce qui impose d'avoir une description "complète" pour chaque unité syntaxique.

Expression_lexicale	:=	Séquence_lexicale Séquence_lexicale + Expression_lexicale	Alternatives possibles.
Séquence_lexicale	:=	Elément_lexical Elément_lexical . Séquence_lexicale	Suite d'éléments lexicaux.
Elément_lexical	:=	(Expression_lexicale) () Unité_lexicale	Différents éléments de base acceptés.

Après avoir fourni les descriptions de base des unités syntaxiques, on modifie, si nécessaire, ces descriptions pour prendre en compte les règles phonologiques. Ces règles, éventuellement contextuelles, consistent à définir des **alternatives** à des prononciations de base (la séquence originale reste possible) ou des **substitutions** portant sur des séquences de phonèmes (la séquence originale est supprimée).

Alternatives { Séquence_en_contexte = Expression_lexicale ; }	Introduction d'alternatives pour des séquences en contexte.
Substitutions { Séquence_en_contexte = Expression_lexicale ; }	Remplacement de séquences en contexte.

Ces deux types de rubriques doivent se trouver nécessairement à la fin des spécifications du niveau lexical. Les règles introduites sont appliquées au fur et à mesure et on peut mettre autant de blocs de chaque type que l'on veut. Pour chacune des occurrences de la séquence, l'expression lexicale est ajoutée en parallèle à la séquence (cas des alternatives) ou bien lui est substituée (cas des substitutions) suivant la nature du bloc courant. L'ajout ou la substitution n'intervient que dans le contexte précisé.

La spécification d'une séquence en contexte revient à décrire une séquence, c'est-à-dire une suite d'unités lexicales et à préciser, si nécessaire, les contextes gauche et droit à considérer. Les unités lexicales de la séquence sont séparés par des "." (points). Les contextes sont définis sous forme d'ensembles d'unités, entre crochets ("[" et "]",) et sont séparés de la séquence par des "." (points).

Séquence_en_contexte	:=	Contexte_gauche . Séquence_lexicale . Contexte_droit Contexte_gauche . Séquence_lexicale Séquence_lexicale . Contexte_droit Séquence_lexicale
Contexte_gauche	:=	[Ensemble_unités_lexicales]

Séquence_lexicale	:=	Unité_lexicale Unité_lexicale . Séquence_lexicale
Contexte_droit	:=	[Ensemble_unités_lexicales]

2.4 Spécifications acoustiques

Le bloc acoustique sert à modéliser, au niveau acoustique, les unités lexicales de la description. Dans le cas où le niveau lexical est omis, les unités lexicales (implicites) correspondent aux unités syntaxiques. Pour obtenir la modélisation acoustique des unités on commence par définir un certain nombre de modèles acoustiques de base et on les affecte ensuite aux différentes unités lexicales. Les modèles acoustiques peuvent comporter plusieurs entrées et sorties auxquelles seront affectés des contextes. Cette approche conduit à une modélisation par allophones, dans le cas où les unités lexicales sont des phonèmes.

Les contextes acoustiques sont des ensembles d'unités lexicales. Leur mode de définition est identique à celui des contextes lexicaux. Par contre, les noms des contextes appartiennent à des listes différentes. On peut donc avoir des contextes lexicaux et acoustiques ayant le même nom mais correspondant à des ensembles différents d'unités lexicales.

2.4.1 Organisation des spécifications

L'ordre normal des rubriques consiste à définir d'abord les **contextes** dont on a besoin, puis les **modèles acoustiques**, et enfin l'affectation des modèles acoustiques aux unités lexicales (la **modélisation** proprement dite).

<pre> Acoustique { Contextes { Nom_contexte = (Unité_lexicale , ...) ; Nom_contexte = [Ensemble_unités] ; } Modele Nom_modèle { Nombre Etats = Nombre_états ; Nombre Transitions = Nombre_transitions ; Nombre Fonctions = Nombre_fonctions ; Etats Entrees = (Nom_état , ...) ; Etats Sorties = (Nom_état , ...) ; Etats Internes = (Nom_état , ...) ; Fonctions = (Nom_fonction , ...) ; Transitions { < Etat_début , Etat_fin , Fonction_associée > ; } } Modelisations { Modèle_acoustique avec { Etat_début = [Ensemble_unités_lexicales] ; Etat_fin = [Ensemble_unités_lexicales] ; } pour Liste_unités_lexicales ; } } </pre>	<p>Déclaration et description des contextes acoustiques.</p> <p>Déclaration et description des modèles acoustiques de base.</p> <p>Affectation des modèles acoustiques aux unités lexicales.</p>
---	--

Cet ordre est relativement libre, l'essentiel étant de définir les objets avant de les utiliser. Au niveau de la définition des modèles acoustiques certaines sous rubriques sont facultatives, mais leur utilisation permet d'effectuer plus de contrôles lors de la compilation du modèle et d'éviter certaines erreurs. En ce qui concerne les affectations, l'utilisation des contextes est facultative; si rien n'est précisé on considère que le contexte ne doit pas être pris en considération et toutes les entrées sorties seront utilisées quelque soit le contexte.

2.4.2 Description des spécifications

La déclaration et la description des **contextes** étant la même qu'au niveau lexical nous ne nous attarderons pas sur ce point, et nous rappellerons juste le bloc de spécification. Signalons quand même que les contextes lexicaux et les contextes acoustiques appartiennent à des listes différentes, et donc qu'un contexte défini au niveau lexical n'est pas connu au niveau acoustique.

<pre>Contextes { Nom_contexte = (Unité_lexicale , ...) ; Nom_contexte = [Ensemble_unités] ; }</pre>	Déclaration et description des contextes acoustiques.
---	---

Le bloc de déclaration et de description des **modèles acoustiques** est le plus important. C'est lui qui décrit la forme du réseau et l'association des fonctions aux transitions. Dans le cas où l'on désire utiliser plusieurs entrées et sorties il faut préciser le nom des états concernés, sinon le modèle ne comportera (par défaut) qu'un seul état d'entrée et un seul état de sortie.

<pre>Modele Nom_modèle { Nombre Etats = Nombre_états ; Nombre Transitions = Nombre_transitions ; Nombre Fonctions = Nombre_fonctions ; Etats Entrees = (Nom_état , ...) ; Etats Sorties = (Nom_état , ...) ; Etats Internes = (Nom_état , ...) ; Fonctions = (Nom_fonction , ...) ; Transitions { < Etat_début , Etat_fin , Fonction_associée > ; } }</pre>	<p>Déclaration de la taille du modèle de base.</p> <p>Déclaration des états et des fonctions à utiliser.</p> <p>Description des transitions.</p>
---	--

Pour les listes d'**états** et de **fonctions**, les noms doivent être séparés par des "," (virgules), et l'ensemble mis entre parenthèses "(" et ")". Si la liste est réduite à un seul élément les parenthèses sont facultatives. Les transitions du réseau sont décrites par des triplets qui indiquent l'état début de la transition, l'état fin ainsi que la fonction qui doit lui être associée.

La spécification d'une transition vide est possible et dépend des déclarations effectuées. Si on a déclaré les fonctions, la transition vide sera indiquée en utilisant une chaîne vide ("") en tant que nom de fonction. Sinon, pour des raisons de compatibilité (avec PHIL86-PC), la fonction nulle est identifiée par le chiffre 0.

Le bloc des **modélisations**, quant à lui, sert à affecter un modèle acoustique de base pour chacune des unités lexicales à modéliser. Lorsque l'on souhaite utiliser des allophones, il faut particulariser les contextes affectés à chaque état d'entrée ou de sortie. Ceci revient à utiliser

des entrées et des sorties différentes suivant les unités lexicales qui suivent ou précèdent. Si pour un état d'entrée ou de sortie, aucun contexte n'est précisé, cet état sera valide quel que soit l'unité lexicale adjacente.

<pre> Modelisations { Modèle_acoustique avec { Etat_début = [Ensemble_unités_lexicales] ; Etat_fin = [Ensemble_unités_lexicales] ; } pour Liste_unités_lexicales ; } </pre>	<p>Nom du modèle à utiliser.</p> <p>Affectation des entrées sorties suivant les contextes.</p> <p>Liste des unités lexicales à modéliser.</p>
---	---

Après le remplacement des unités par les modèles spécifiés, on vérifie qu'il existe bien des chemins permettant de relier chacune des entrées utilisées à chacune des sorties utilisées, compte tenu des contextes possibles.

3. Fichier bibliothèque (.XDF)

Lorsque l'on spécifie le modèle à utiliser pour une application de reconnaissance de la parole, on introduit des connaissances a priori. Un certain nombre d'entre elles sont indépendantes de l'application. En conséquence, il est inutile de les répéter (recopier) à chaque fois et il vaut mieux les mettre dans un fichier particulier, qualifié de fichier bibliothèque.

Ce fichier, facultatif, comporte deux blocs principaux. Le premier bloc est identifié par l'un des deux mots clés "Avant" ou "Initialisations" (ce sont des synonymes). Il contient principalement des initialisations, c'est-à-dire des déclarations d'unités et des descriptions, d'où le second mot clé; et il est traité avant le fichier principal, d'où le premier mot clé. Le deuxième bloc est identifié par l'un des mots clés "Après" ou "Modifications". Ce sont encore deux synonymes correspondant au fait que ce bloc est traité après le fichier principal et qu'il comporte principalement des modifications (substitutions et alternatives).

Si on déclare des unités (ou éléments) d'un type donné en bibliothèque, il faudra également déclarer toutes les autres unités du même type apparaissant dans le fichier principal. Le décodage du bloc "Avant" de la bibliothèque sert principalement à fournir un certain nombre de valeurs par défaut, et donc à initialiser les listes. ***La différence principale entre le fichier bibliothèque et le fichier principal, est que l'on n'a pas de "warnings" si l'on n'utilise pas toutes les informations déclarées ou décrites en bibliothèque.***

Le rôle et la description des rubriques étant identiques entre le fichier bibliothèque et le fichier principal, nous nous limiterons à mentionner les rubriques possibles dans chacun des deux blocs principaux. Rappelons quand même que le programme de compilation décode les informations fournies dans les fichiers (fichier principal et fichier bibliothèque) dans l'ordre suivant :

- bloc des "Initialisations" (ou bloc "Avant") du fichier bibliothèque,
- le fichier principal (blocs syntaxique, lexical et acoustique),
- et enfin le bloc des "Modifications" (ou bloc "Après") du fichier bibliothèque.

3.1 Bloc des initialisations

Le bloc des initialisations du fichier bibliothèque est identifié soit par le mot clé "Avant", soit par le clé "Initialisations" (au choix). Son rôle principal est de permettre la déclaration des unités lexicales, la définition de contextes lexicaux et acoustiques, ainsi que la spécification d'un jeu de modèles acoustiques de base.

<pre> Avant { Syntaxe { ... } Lexique { ... } Acoustique { ... } }</pre>	Initialisations syntaxiques. Initialisations lexicales. Initialisations acoustiques.
--	---

Attention, bien que les objets déclarés ou décrits en bibliothèque ne soient pas cause de "warnings" s'ils ne sont pas utilisés, ils occupent néanmoins de la place en mémoire, et leur décodage prend du temps. Il est donc préférable de ne pas définir trop d'objets inutiles.

3.1.1 Spécifications syntaxiques

Les seules rubriques acceptées au niveau syntaxique concernent la déclaration et la description d'éléments terminaux ou non terminaux. Il n'est pas possible de définir des entrées syntaxiques (points d'entrée du réseau principal).

<pre> Syntaxe { Elements Terminaux = (Elément_terminal , ...) ; Elements Non_Terminaux = (Elément_non_terminal , ...) ; Elements Terminaux { Elément_terminal = < Mot_syntaxique , Unité_syntaxique > ; } Elements Non_Terminaux { Elément_non_terminal = Expression_syntaxique ; } } </pre>	<p>Déclaration des éléments syntaxiques.</p> <p>Déclaration des sous-syntaxes.</p> <p>Déclaration et description des éléments syntaxiques.</p> <p>Déclaration et description des sous-syntaxes.</p>
---	---

Si, dans le fichier bibliothèque, on déclare quelques éléments terminaux, ou non terminaux, ou si on trouve simplement les mots clés correspondants, il faudra ensuite déclarer tous les autres éléments du même type qui seront utilisés dans le fichier principal.

3.1.2 Spécifications lexicales

Au niveau lexical, on retrouve plus de rubriques, les deux principales étant la déclaration des unités lexicales et la définition de contextes lexicaux en vue de l'application de règles contextuelles.

<pre> Lexique { Unites = (Unité_lexicale , ...) ; Contextes { Nom_contexte = (Unité_lexicale , ...) ; Nom_contexte = [Ensemble_unités] ; } Descriptions { Unité_syntaxique = Expression_lexicale ; } } </pre>	<p>Déclaration des unités lexicales.</p> <p>Déclaration et description de contextes lexicaux.</p> <p>Description d'unités syntaxiques.</p>
--	--

La principale utilisation de ces rubriques est de permettre la déclaration des unités lexicales, ce qui entraîne le contrôle de toutes les unités apparaissant dans les descriptions lexicales.

3.1.3 Spécifications acoustiques

On niveau acoustique, on peut également définir un certain nombre de contextes ainsi qu'un ensemble de modèles acoustiques de base.

<pre> Acoustique { Contextes { Nom_contexte = (Unité_lexicale , ...) ; Nom_contexte = [Ensemble_unités] ; } Modele Nom_modèle { Nombre Etats = Nombre_états ; Nombre Transitions = Nombre_transitions ; Nombre Fonctions = Nombre_fonctions ; Etats Entrees = (Nom_état , ...) ; Etats Sorties = (Nom_état , ...) ; Etats Internes = (Nom_état , ...) ; Fonctions = (Nom_fonction , ...) ; Transitions { < Etat_début , Etat_fin , Fonction_associée > ; } } } </pre>	<p>Déclaration et description de contextes acoustiques.</p> <p>Déclaration et description des modèles acoustiques de base.</p>
---	--

Une des principales utilisations de la bibliothèque, à ce niveau, consistera à définir un ensemble de modèles acoustiques de différentes tailles (modélisation par mots) ou de différentes formes (modélisation par allophones). Ces modèles occupent, au moins temporairement, de la place en mémoire et leur décodage prend du temps. Il faut donc faire un compromis au moment de la définition de la bibliothèque, c'est-à-dire ne pas faire d'excès.

3.2 Bloc des modifications

Le bloc des modifications du fichier bibliothèque est identifié soit par le mot clé "**Apres**", soit par le clé "**Modifications**" (au choix). Son rôle principal est de permettre l'application d'un jeu de règles phonologiques ainsi que la mise en oeuvre d'une modélisation contextuelle (allophones).

<pre> Apres { Syntaxe { ... } Lexique { ... } Acoustique { ... } } </pre>	<p>Modifications syntaxiques.</p> <p>Modifications lexicales.</p> <p>Modifications acoustiques.</p>
--	---

Toutes ces rubriques seront traitées après le fichier principal. Les règles de substitution et les alternatives ne seront pas causes de "warnings" si elles ne sont pas utilisées. Toutefois, leur application peut prendre pas mal de temps (dépend de la taille des descriptions à traiter). Par contre, les règles ne consomment pas beaucoup de place mémoire car elles sont appliquées au fur et à mesure qu'elles sont rencontrées.

3.2.1 Spécifications syntaxiques

Au niveau syntaxique on peut trouver les rubriques de substitutions de mots ou d'unités syntaxiques.

<pre> Syntaxe { Substitutions Mots { Ancien_mot_syntaxique = Nouveau_mot_syntaxique ; } Substitutions Unites { Ancienne_unité_syntaxique = Nouvelle_unité_syntaxique ; } } </pre>	<p>Modifications portant sur les mots syntaxiques.</p> <p>Modifications portant sur les unités syntaxiques.</p>
---	---

Ces blocs sont disponibles, mais en fait peu utiles car la partie syntaxique constitue l'ensemble d'informations qui est le plus spécifique de chaque application.

3.2.2 Spécifications lexicales

Au niveau lexical, on trouvera principalement des règles phonologiques ainsi que la définition de contextes si ceux-ci n'ont pas été mis dans le bloc des initialisations.

Ce bloc permet l'introduction d'un ensemble de règles phonologiques qui seront appliquées à l'ensemble des descriptions lexicales. On peut avoir autant de blocs de chaque type que l'on veut, et ils peuvent apparaître dans n'importe quel ordre. La définition des contextes est disponible à ce niveau de la bibliothèque afin de permettre à chacun de choisir sa méthode de déclaration préférée : soit une déclaration et une description préalable de tous les contextes nécessaires, soit une déclaration au coup par coup au fur et à mesure des besoins.

<pre> Lexique { Contextes { Nom_contexte = (Unité_lexicale , ...) ; Nom_contexte = [Ensemble_unités] ; } Alternatives { Séquence_en_contexte = Expression_lexicale ; } Substitutions { Séquence_en_contexte = Expression_lexicale ; } } </pre>	<p>Déclaration et définition de contextes lexicaux.</p> <p>Alternatives pour des séquences en contexte.</p> <p>Remplacement de séquences en contexte.</p>
--	---

3.2.3 Spécifications acoustiques

Au niveau acoustique, on trouvera les règles de modélisation des unités lexicales. Ce bloc servant principalement dans le cas d'une modélisation par allophones.

<pre> Acoustique { Contextes { Nom_contexte = (Unité_lexicale , ...) ; Nom_contexte = [Ensemble_unités] ; } Modelisations { Modèle_acoustique avec { Etat_début = [Ensemble_unités_lexicales] ; Etat_fin = [Ensemble_unités_lexicales] ; } pour Liste_unités_lexicales ; } } </pre>	<p>Déclaration et description de contextes acoustiques.</p> <p>Affectation des modèles acoustiques aux unités lexicales.</p>
---	--

Ici, comme au niveau lexical, on peut choisir entre une déclaration préalable de l'ensemble des contextes nécessaires, ou bien une déclaration au fur et à mesure des besoins. La partie modélisation est sans doute la partie la plus critique, mais aussi la plus utile en bibliothèque dans le cas des modélisations par allophones.

4. Syntaxe des spécifications

Ce chapitre regroupe la syntaxe qui doit être respectée lors de la spécification du modèle pour une application de reconnaissance. Ces spécifications sont regroupées par niveaux et types de rubriques, et les mots clés sont indiqués en gras. Les séparateurs (signes de ponctuation) apparaissant dans les instructions sont obligatoires.

Rappelons que le décodage des informations fournies se fait dans l'ordre suivant : d'abord le bloc d'**initialisations** du fichier bibliothèque, puis le **fichier principal**, et enfin le bloc de **modifications** du fichier bibliothèque. Pour chaque niveau de données (syntaxique, lexical et acoustique), on donne la liste des rubriques qui peuvent être spécifiées dans le fichier de bibliothèque, soit dans le cadre des initialisations (bloc **Initialisations** { ... } ou **Avant** { ... }), soit dans le cadre des modifications (bloc **Modifications** { ... } ou **Après** { ... }).

4.1 Contenu des fichiers

Spécifications dans le fichier principal de définition (.DEF)

Spécifications_complètes	:=	Modélisation_par_mots Modélisation_par_allophones
Modélisation_par_mots	:=	Syntaxe { Spécifications_syntaxiques } Acoustique { Spécifications_acoustiques }
Modélisation_par_allophones	:=	Syntaxe { Spécifications_syntaxiques } Lexique { Spécifications_lexicales } Acoustique { Spécifications_acoustiques }

Spécifications dans le fichier bibliothèque éventuel (.XDF)

Spécifications_bibliothèque	:=	Spécifications_avant_fichier Spécifications_après_fichier
Spécifications_avant_fichier	:=	Avant { Spécifications_initialisations } Initialisations { Spécifications_initialisations }
Spécifications_après_fichier	:=	Après { Spécifications_modifications } Modifications { Spécifications_modifications }
Spécifications_initialisations	:=	Syntaxe { Initialisations_syntaxiques } Lexique { Initialisations_lexicales } Acoustique { Initialisations_acoustiques }
Spécifications_modifications	:=	Syntaxe { Modifications_syntaxiques } Lexique { Modifications_lexicales } Acoustique { Modifications_acoustiques }

4.2 Spécifications syntaxiques

Initialisations syntaxiques dans le fichier bibliothèque

Initialisations_syntaxiques	:=	Initialisation_syntaxique Initialisation_syntaxique Initialisations_syntaxiques
Initialisation_syntaxique	:=	Elements Terminaux = Liste_terminaux ; Elements Non Terminaux = Liste_non_terminaux ; Elements Terminaux { Spécifications_terminaux } Elements Non Terminaux { Spécifications_non_terminaux }

Spécifications syntaxiques dans le fichier principal de définition

Spécifications_syntaxiques	:=	Informations_syntaxiques Modifications_syntaxiques
Informations_syntaxiques	:=	Information_syntaxique Information_syntaxique Informations_syntaxiques
Modifications_syntaxiques	:=	Modification_syntaxique Modification_syntaxique Modifications_syntaxiques
Information_syntaxique	:=	Elements Terminaux = Liste_terminaux ; Elements Non Terminaux = Liste_non_terminaux ; Entrees Syntaxiques = Liste_entrées_syntaxiques ; Productions { Descriptions_productions } Elements Terminaux { Spécifications_terminaux } Elements Non Terminaux { Spécifications_non_terminaux } Entrees Syntaxiques { Spécifications_entrées_syntaxiques }
Modification_syntaxique	:=	Substitutions Mots { Règles_substitutions_mots } Substitutions Unites { Règles_substitutions_unités }

Modifications syntaxiques dans le fichier bibliothèque

Modifications_syntaxiques	:=	Modification_syntaxique Modification_syntaxique Modifications_syntaxiques
Modification_syntaxique	:=	Substitutions Mots { Règles_substitutions_mots } Substitutions Unites { Règles_substitutions_unités }

Déclarations des éléments syntaxiques

Liste_terminaux	:=	Elément_terminal (Suite_éléments_terminaux)
Suite_éléments_terminaux	:=	Elément_terminal Elément_terminal , Suite_éléments_terminaux
Liste_non_terminaux	:=	Elément_non_terminal (Suite_éléments_non_terminaux)
Suite_éléments_non_terminaux	:=	Elément_non_terminal Elément_non_terminal , Suite_éléments_non_terminaux
Liste_entrées_syntaxiques	:=	Entrée_syntaxique (Suite_entrées_syntaxiques)
Suite_entrées_syntaxiques	:=	Entrée_syntaxique Entrée_syntaxique , Suite_entrées_syntaxiques

Description des productions syntaxiques (sans déclaration)

Descriptions_productions	:=	Description_production Description_production Descriptions_productions
Description_production	:=	Elément_non_terminal = Expression_syntaxique ; Entrée_syntaxique = Expression_syntaxique ;

Déclarations et descriptions des éléments syntaxiques

Spécifications_terminaux	:=	Spécification_terminal Spécification_terminal Spécifications_terminaux
Spécification_terminal	:=	Elément_terminal ; Elément_terminal = Couple_syntaxique ;
Couple_syntaxique	:=	< Mot_syntaxique , Unité_syntaxique > < Mot_syntaxique , * > < * , Unité_syntaxique > < * , * >
Spécifications_non_terminaux	:=	Spécification_non_terminal Spécification_non_terminal Spécifications_non_terminaux
Spécification_non_terminal	:=	Elément_non_terminal ; Elément_non_terminal = Expression_syntaxique ;
Spécifications_entrées_syntaxiques	:=	Spécification_entrée_syntaxique Spécification_entrée_syntaxique Spécifications_entrées_syntaxiques
Spécification_entrée_syntaxique	:=	Entrée_syntaxique ; Entrée_syntaxique = Expression_syntaxique ;

Spécification des expressions syntaxiques (=> réseaux syntaxiques)

Expression_syntaxique	:=	Séquence_syntaxique Séquence_syntaxique + Expression_syntaxique
Séquence_syntaxique	:=	Boucle_syntaxique Boucle_syntaxique . Séquence_syntaxique
Boucle_syntaxique	:=	Elément_syntaxique Elément_syntaxique *
Elément_syntaxique	:=	() (Expression_syntaxique) Elément_non_terminal Elément_terminal Couple_syntaxique
Couple_syntaxique	:=	< Mot_syntaxique , Unité_syntaxique >

Modifications des descriptions (réseaux) syntaxiques

Règles_substitutions_mots	:=	Règle_substitution_mot Règle_substitution_mot Règles_substitutions_mots
Règle_substitution_mot	:=	Ancien_mot_syntaxique = Nouveau_mot_syntaxique ;
Ancien_mot_syntaxique	:=	Mot_syntaxique
Nouveau_mot_syntaxique	:=	Mot_syntaxique
Règles_substitutions_unités	:=	Règle_substitution_unité Règle_substitution_unité Règles_substitutions_unités
Règle_substitution_unité	:=	Ancienne_unité_syntaxique = Nouvelle_unité_syntaxique ;
Ancienne_unité_syntaxique	:=	Unité_syntaxique
Nouvelle_unité_syntaxique	:=	Unité_syntaxique

4.3 Spécifications lexicales**Initialisations lexicales dans le fichier bibliothèque**

Initialisations_lexicales	:=	Initialisation_lexicale Initialisation_lexicale Initialisations_lexicales
Initialisation_lexicale	:=	Unites = Liste_unités_lexicales ; Contextes { Spécifications_contextes } Descriptions { Spécifications_descriptions }

Spécifications lexicales dans le fichier principal de définition

Spécifications_lexicales	:=	Informations_lexicales Modifications_lexicales
Informations_lexicales	:=	Information_lexicale Information_lexicale Informations_lexicales
Modifications_lexicales	:=	Modification_lexicale Modification_lexicale Modifications_lexicales
Information_lexicale	:=	Unites = Liste_unités_lexicales ; Contextes { Spécifications_contextes } Descriptions { Spécifications_descriptions }
Modification_lexicale	:=	Contextes { Spécifications_contextes } Alternatives { Spécifications_alternatives } Substitutions { Spécifications_substitutions }

Modifications lexicales dans le fichier bibliothèque

Modifications_lexicales	:=	Modification_lexicale Modification_lexicale Modifications_lexicales
Modification_lexicale	:=	Contextes { Spécifications_contextes } Alternatives { Spécifications_alternatives } Substitutions { Spécifications_substitutions }

Déclaration des unités lexicales (en général des phonèmes)

Liste_unités_lexicales	:=	Unité_lexicale (Suite_unités_lexicales)
Suite_unités_lexicales	:=	Unité_lexicale Unité_lexicale , Suite_unités_lexicales

Déclaration et description de contextes au niveau lexical (pour règles contextuelles)

Spécifications_contextes	:=	Spécification_contexte Spécification_contexte Spécifications_contextes
Spécification_contexte	:=	Nom_contexte = Liste_unités_lexicales ; Nom_contexte = Contexte_lexical ;

Description des unités syntaxiques (=> expressions lexicales)

Spécifications_descriptions	:=	Spécification_description Spécification_description Spécifications_descriptions
Spécification_description	:=	Unité_syntaxique = Expression_lexicale ;

Spécification des expressions lexicales (expressions phonétiques)

Expression_lexicale	:=	Séquence_lexicale Séquence_lexicale + Expression_lexicale
Séquence_lexicale	:=	Élément_lexical Élément_lexical . Séquence_lexicale
Élément_lexical	:=	(Expression_lexicale) () Unité_lexicale

Modifications des descriptions lexicales (réseaux lexicaux)

Spécifications_alternatives	:=	Spécification_alternative Spécification_alternative Spécifications_alternatives
Spécification_alternative	:=	Séquence_en_contexte = Expression_à_ajouter ;
Expression_à_ajouter	:=	Expression_lexicale
Spécifications_substitutions	:=	Spécification_substitution Spécification_substitution Spécifications_substitutions
Spécification_substitution	:=	Séquence_en_contexte = Expression_à_substituer ;
Expression_à_substituer	:=	Expression_lexicale

Spécification de séquences lexicales (phonétiques) en contexte

Séquence_en_contexte	:	= Contexte_gauche . Séquence_lexicale . Contexte_gauche . Séquence_lexicale Séquence_lexicale . Contexte_droit Séquence_lexicale
Contexte_gauche	:	= Contexte_lexical
Séquence_lexicale	:	= Unité_lexicale Unité_lexicale . Séquence_lexicale
Contexte_droit	:	= Contexte_lexical

Spécification de contextes (ensembles d'unités lexicales)

Contexte_lexical	:=	[Ensemble_unités]
Ensemble_unités	:=	* Nom_contexte Unité_lexicale Ensemble_unités + Nom_contexte Ensemble_unités - Nom_contexte Ensemble_unités + Unité_lexicale Ensemble_unités - Unité_lexicale

4.4 Spécifications acoustiques

Initialisations acoustiques dans le fichier bibliothèque

Initialisations_acoustiques	:=	Initialisation_acoustique Initialisation_acoustique Initialisations_acoustiques
Initialisation_acoustique	:=	Contextes { Spécifications_contextes } Modele Nom_modèle { Spécifications_modèle }

Spécifications acoustiques dans le fichier principal de définition

Spécifications_acoustiques	:=	Spécification_acoustique Spécification_acoustique Spécifications_acoustiques
Spécification_acoustique	:=	Contextes { Spécifications_contextes } Modele Nom_modèle { Spécifications_modèle } Modélisations { Spécifications_modélisations }

Modifications acoustiques dans le fichier bibliothèque

Modifications_acoustiques	:=	Modification_acoustique Modification_acoustique Modifications_acoustiques
Modification_acoustique	:=	Contextes { Spécifications_contextes } Modélisations { Spécifications_modélisations }

Déclaration et description de contextes au niveau acoustique (pour les modélisations contextuelles)

Spécifications_contextes	:=	Spécification_contexte Spécification_contexte Spécifications_contextes
Spécification_contexte	:=	Nom_contexte = Liste_unités_lexicales ; Nom_contexte = Contexte_acoustique ;

Déclaration et description des modèles acoustiques

Spécifications_modèle	:=	Spécification_modèle Spécification_modèle Spécifications_modèle
Spécification_modèle	:=	Nombre Etats = Nombre_états ; Nombre Transitions = Nombre_transitions ; Nombre Fonctions = Nombre_fonctions ; Etats Entrees = Liste_états ; Etats Sorties = Liste_états ; Etats Internes = Liste_états ; Fonctions = Liste_fonctions ; Transitions { Spécifications_transitions }

Spécification des modélisations des unités lexicales

Spécifications_modélisations	:=	Spécification_modélisation Spécification_modélisation Spécifications_modélisations
Spécification_modélisation	:=	Modélisation_hors_contexte Modélisation_contextuelle
Modélisation_hors_contexte	:=	Modèle_acoustique pour Liste_unités_lexicales ;
Modélisation_contextuelle	:=	Modèle_acoustique avec { Entrées_sorties } pour Liste_unités_lexicales ;

Spécification de contextes (ensembles d'unités lexicales, idem lexique)

Contexte_acoustique	:=	[Ensemble_unités]
Ensemble_unités	:=	* Nom_contexte Unité_lexicale Ensemble_unités + Nom_contexte Ensemble_unités - Nom_contexte Ensemble_unités + Unité_lexicale Ensemble_unités - Unité_lexicale

Déclaration des états et des fonctions des modèles acoustiques

Liste_états	:=	Nom_état (Suite_noms_états)
Suite_noms_états	:=	Nom_état Nom_état , Suite_noms_états
Liste_fonctions	:=	Nom_fonction (Suite_noms_fonctions)
Suite_noms_fonctions	:=	Nom_fonction Nom_fonction , Suite_noms_fonctions

Spécification des transitions des modèles acoustiques

Spécifications_transitions	:=	Spécification_transition Spécification_transition Spécifications_transitions
Spécification_transition	:=	< Etat_début , Etat_fin , Fonction_associée > ;
Etat_début	:=	Nom_état
Etat_fin	:=	Nom_état
Fonction_associée	:=	Nom_fonction

Affectation de contextes acoustiques lors de la modélisation des unités

Entrées_sorties	:=	Entrée_sortie Entrée_sortie Entrées_sorties
Entrée_sortie	:=	Etat_début = Contexte_acoustique ; Etat_fin = Contexte_acoustique ;
Modèle_acoustique	:=	Nom_modèle
Etat_début	:=	Nom_état
Etat_fin	:=	Nom_état

4.5 Eléments de base***Symboles apparaissant dans les spécifications du niveau syntaxique***

Entrée_syntaxique	:=	Chaîne_de_caractères
Elément_non_terminal	:=	Chaîne_de_caractères
Elément_terminal	:=	Chaîne_de_caractères
Mot_syntaxique	:=	Chaîne_de_caractères Chaîne_vide
Unité_syntaxique	:=	Chaîne_de_caractères

Symboles apparaissant dans les spécifications du niveau lexical

Unité_syntaxique	:=	Chaîne_de_caractères
Unité_lexicale	:=	Chaîne_de_caractères
Nom_contexte	:=	Chaîne_de_caractères

Symboles apparaissant dans les spécifications du niveau acoustique

Unité_lexicale	:=	Chaîne_de_caractères
Nom_contexte	:=	Chaîne_de_caractères
Nom_modèle	:=	Chaîne_de_caractères
Nom_état	:=	Symbole_alphanumérique
Nom_fonction	:=	Symbole_alphanumérique Chaîne_vide
Nombre_états	:=	Valeur_numérique_entière
Nombre_transitions	:=	Valeur_numérique_entière
Nombre_fonctions	:=	Valeur_numérique_entière

Différents types d'éléments de base

Chaîne_de_caractères	:=	Symbole_alphanumérique "Suite_de_caractères" Chaîne_vide
Valeur_numérique_entière	:=	<i>Suite de caractères numériques : 0 ... 9.</i>
Séparateurs	:=	<i>La plupart des caractères ascii qui ne sont pas alphanumériques, par exemple : . , ; = + - * () { } < > ! ... Attention, tous les séparateurs n'ont pas nécessairement de signification syntaxique.</i>
Chaîne_vide	:=	<i>" " (deux doubles quotes)</i>
Symbole_alphanumérique	:=	<i>Suite de caractères alphanumériques : 0 ... 9, a ... z, A ... Z, _ (souligné) et \$ (dollar). Ces chaînes de caractères sont délimitées par des caractères d'espacement ou des séparateurs.</i>
Suite_de_caractères	:=	<i>Suite de caractères quelconques : caractères alphanumériques, caractères d'espacement, caractères accentués (code supérieur à 128), etc. Toutes les séquences avec \... sont interdites exceptés \\ qui donne \ et \" qui donne "</i>

5. Schéma des spécifications

Ces quelques pages résument le contenu des fichiers de spécification d'une application : fichier principal et fichier bibliothèque. Ce résumé permet de retrouver rapidement l'ordre standard des rubriques ainsi que les mots clés reconnus par le programme de compilation.

5.1 Fichier principal de définition (.DEF)

```
Syntaxe {  
  Elements Terminaux = ( Elément_terminal , ... ) ;  
  Elements Non_Terminaux = ( Elément_non_terminal , ... ) ;  
  Entrees Syntaxiques = ( Entrée_syntaxique , ... ) ;  
  Productions {  
    Entrée_syntaxique = Expression_syntaxique ;  
    Elément_non_terminal = Expression_syntaxique ;  
  }  
  Elements Terminaux {  
    Elément_terminal = < Mot_syntaxique , Unité_syntaxique > ;  
  }  
  Elements Non_Terminaux {  
    Elément_non_terminal = Expression_syntaxique ;  
  }  
  Entrees Syntaxiques {  
    Entrée_syntaxique = Expression_syntaxique ;  
  }  
  Substitutions Mots {  
    Ancien_mot_syntaxique = Nouveau_mot_syntaxique ;  
  }  
  Substitutions Unites {  
    Ancienne_unité_syntaxique = Nouvelle_unité_syntaxique ;  
  }  
}  
  
! Dans les expressions syntaxiques  
!   + indique un choix (une alternative),  
!   . une concaténation,  
!   * un bouclage  
!   et ( ... ) un regroupement (sous expression syntaxique).
```

```

Lexique {

  Unités = ( Unité_lexicale , ... ) ;

  Contextes {
    Contexte_lexical = ( Unité_lexicale , ... ) ;
    Contexte_lexical = [ Ensemble_unités_lexicales ] ;
  }

  Descriptions {
    Unité_syntactique = Expression_lexicale ;
  }

  Alternatives {
    [ contexte_gauche ] . suite_unités . [ contexte_droit ]
                          = Expression_lexicale_à_ajouter ;
  }

  Substitutions {
    [ contexte_gauche ] . suite_unités . [ contexte_droit ]
                          = Expression_lexicale_à_substituer ;
  }
}

! Dans les expressions lexicales
!   + indique un choix (une alternative),
!   . une concaténation
!   et ( ... ) un regroupement (sous expression lexicale).

```

```

Acoustique {

  Contextes {
    Contexte_acoustique = ( Unité_lexicale , ... ) ;
    Contexte_acoustique = [ Ensemble_unités_lexicales ] ;
  }

  Modele Nom_modèle_acoustique {
    Nombre Etats = Nombre_états ;
    Nombre Transitions = Nombre_transitions ;
    Nombre Fonctions = Nombre_fonctions ;
    Etats Entrees = ( Nom_état , ... ) ;
    Etats Sorties = ( Nom_état , ... ) ;
    Etats Internes = ( Nom_état , ... ) ;
    Fonctions = ( Nom_fonction , ... ) ;
    Transitions {
      < Etat_début , Etat_fin , Fonction_associée > ;
    }
  }

  Modelisations {
    Nom_modèle_acoustique
    avec {
      Etat_début = [ Ensemble_unités_lexicales ] ;
      Etat_fin = [ Ensemble_unités_lexicales ] ;
    } pour ( Unité_lexicale , ... ) ;
  }
}

```

5.2 Fichier bibliothèque éventuel (.XDF)

Initialisations {	! ou bien identificateur Avant {
Syntaxe { Elements Terminaux = (Elément_terminal , ...) ; Elements Non_Terminaux = (Elément_non_terminal , ...) ; Elements Terminaux { Elément_terminal = < Mot_syntaxique , Unité_syntaxique > ; } Elements Non_Terminaux { Elément_non_terminal = Expression_syntaxique ; } }	
Lexique { Unités = (Unité_lexicale , ...) ; Contextes { Contexte_lexical = (Unité_lexicale , ...) ; Contexte_lexical = [Ensemble_unités_lexicales] ; } Descriptions { Unité_syntaxique = Expression_lexicale ; } }	
Acoustique { Contextes { Contexte_acoustique = (Unité_lexicale , ...) ; Contexte_acoustique = [Ensemble_unités_lexicales] ; } Modele Nom_modèle_acoustique { Nombre Etats = Nombre_états ; Nombre Transitions = Nombre_transitions ; Nombre Fonctions = Nombre_fonctions ; Etats Entrees = (Nom_état , ...) ; Etats Sorties = (Nom_état , ...) ; Etats Internes = (Nom_état , ...) ; Fonctions = (Nom_fonction , ...) ; Transitions { < Etat_début , Etat_fin , Fonction_associée > ; } } }	
}	

Modifications {	! ou bien identificateur Apres {
Syntaxe { Substitutions Mots { Ancien_mot_syntaxique = Nouveau_mot_syntaxique ; } Substitutions Unites { Ancienne_unité_syntaxique = Nouvelle_unité_syntaxique ; } } 	
Lexique { Contextes { Contexte_lexical = (Unité_lexicale , ...) ; Contexte_lexical = [Ensemble_unités_lexicales] ; } Alternatives { Séquence_lexicale_en_contexte = Autre_expression_lexicale_possible ; } Substitutions { Séquence_lexicale_en_contexte = Expression_lexicale_a_substituer ; } } 	
Acoustique { Contextes { Contexte_acoustique = (Unité_lexicale , ...) ; Contexte_acoustique = [Ensemble_unités_lexicales] ; } Modelisations { Nom_modèle_acoustique avec { Etat_début = [Ensemble_unités_lexicales] ; Etat_fin = [Ensemble_unités_lexicales] ; } pour (Unité_lexicale , ...) ; } } 	
}	

6. Remarques et limitations

On regroupe ici quelques remarques concernant les principales limitations sur la spécification et le développement des modèles.

Dans le cas où on utilise des modélisations acoustiques contextuelles, il ne faut pas spécifier de restrictions de contextes pour les entrées des unités lexicales débutant les phrases (silence de début en général) ni pour les sorties des unités lexicales terminant les phrases (silences de fin en général).

Le traitement des règles et modélisations contextuelles nécessite la manipulation de contextes. Au point de vue occupation mémoire il est préférable de définir un certain nombre de contextes, puis de les référencer par leur nom. En effet l'utilisation d'un seul et unique contexte sous la forme [Nom_Contexte] permet une manipulation de pointeur, et supprime la copie de l'ensemble des unités.