UNIVERSITAS INDONESIA

UNIVERSITE DE BRETAGNE SUD

REKAYASA ULANG SISTEM PENDETEKSI EMOSI UNTUK ROBOT PENDAMPING ANAK DI RUMAH SAKIT RAWAT INAP

TESIS

AGUS MULYANTO
1006803884

FAKULTAS TEKNIK
PROGRAM STUDI TEKNIK ELEKTRO
DEPOK
JULI 2012

UNIVERSITAS INDONESIA **UNIVERSITE DE BRETAGNE SUD**

REKAYASA ULANG SISTEM PENDETEKSI EMOSI UNTUK ROBOT PENDAMPING ANAK DI RUMAH SAKIT RAWAT INAP

TESIS

Diajukan sebagai salah satu syarat untuk memperoleh gelar Magister Teknik

AGUS MULYANTO
1006803884

FAKULTAS TEKNIK
PROGRAM STUDI TEKNIK ELEKTRO
KEKHUSUSAN JARINGAN INFORMASI DAN MULTIMEDIA
DEPOK
JULI 2012

KATA PENGANTAR

Puji syukur saya panjatkan kepada Tuhan Yang Maha Esa, karena atas berkat dan rahmat-Nya, saya dapat menyelesaikan tesis ini. Penulisan tesis ini dilakukan dalam rangka memenuhi salah satu syarat untuk mencapai gelar magister Teknik program studi Teknik Elektro pada Fakultas Teknik Universitas Indonesia. Saya menyadari bahwa, tanpa bantuan dan bimbingan dari berbagai pihak, dari masa perkuliahan sampai pada penyusunan tesis ini, sangatlah sulit bagi saya untuk menyelesaikan tesis ini. Oleh karena itu, saya mengucapkan terima kasih kepada:

1) Ayah, Ibu, dan keluarga yang selalu memberikan semangat, do'a serta dukungan setiap saat.
2) Bapak Dr. H.M. Nasrullah Yusuf, S.E., M.B.A. selaku ketua STMIK Teknokrat yang telah memberikan kesempatan kepada penulis untuk melanjutkan studi ke jenjang Master.
3) Dr. Jeanne Villaneau dan Dr. Jean Yves Antoine, selaku dosen pembimbing yang telah menyediakan waktu, tenaga, dan pikiran untuk mengarahkan saya dalam penyusunan tesis ini.
4) Lab-IRISA université de bretagne sud yang telah menyediakan kesempatan untuk melakukan penelitian dan perangkat yang mendukung penelitian ini;
5) *Some one* yang selalu memberikan dorongan dan semangat yang tak kenal waktu, jarak dan kondisi .
6) Dan Semua sahabat yang tidak dapat penulis sebutkan satu persatu, yang telah membantu dalam menyelesaikan tesis ini.

Akhir kata, saya berharap Tuhan Yang Maha Esa berkenan membalas segala kebaikan semua pihak yang telah membantu. Semoga tesis ini membawa manfaat bagi pengembangan ilmudan pendidikan.

Vannes, 27 Juni 2012
Agus Mulyanto

Rekayasa ulang..., Agus Mulyanto, FT UI, 2012

Internship Report

In order to obtain the Master 2 Informatique des Images et des Réseaux (IIR)

Faculty of Sciences and Engineering Sciencess

Université de Bretagne-Sud

Have been presented and defended by:

AGUS MULYANTO

On June 27, 2012

Title:

REENGINEERING OF A EMOTION DETECTION SYSTEM FOR ROBOTS COMPANION WHO AIMED TO CHILDREN IN THE INPATIENTS HOSPITAL

Supervisors:

Dr. Jeanne VILLANEAU :

Dr. Jean Yves ANTOINE :

Examiners:

Prof. Franck POIRIER :

Dr. Gildas MÉNIER :

Vannes, 27 June 2012
Responsible Master 2 IIR

Dr. Frédéric GUIDEC

UNIVERSITE DE BRETAGNE-SUD
UFR Sciences et Sciences de l'Ingénieur

Rekayasa ulang..., Agus Mulyanto, FT UI, 2012

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIK

Sebagai sivitas akademik Universitas Indonesia, Saya yang bertanda tangan di bawah ini :

Nama	: Agus Mulyanto
NPM	: 1006803884
Program Studi	: JARINGAN INFORMASI dan MULTIMEDIA
Departemen	: TEKNIK ELEKTRO
Fakultas	: TEKNIK
Jenis karya	: TESIS

demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Indonesia **Hak Bebas Royalti Noneksklusif (*Non-exclusive Royalty-Free Right*)** atas karya ilmiah saya yang berjudul :

Rekayasa Ulang Sistem Pendeteksi Emosi untuk Robot Pendamping Anak di Rumah Sakit Rawat Inap

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini Universitas Indonesia berhak menyimpan, mengalihmedia/formatkan, mengelola dalam bentuk pangkalan data (*database*), merawat, dan memublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Vannes

Pada tanggal : 27 Juni 2012

Yang menyatakan

(Agus Mulyanto)

Rekayasa ulang..., Agus Mulyanto, FT UI, 2012

HALAMAN PERNYATAAN ORISINALITAS

Tesis ini adalah hasil karya saya sendiri,
dan semua sumber baik yang dikutip maupun dirujuk
telah saya nyatakan dengan benar.

Nama	**: Agus Mulyanto**
NPM	**: 1006803884**
Tanda Tangan	**:**
Tanggal	**: 27 JUNI 2012**

Rekayasa ulang..., Agus Mulyanto, FT UI, 2012

ABSTRACT

Nama : Agus Mulyanto

Program Studi : Teknik Elektro

Judul : Rekayasa Ulang Sistem Pendeteksi Emosi untuk Robot Pendamping Anak di Rumah Sakit Rawat Inap

Sistem pendeteksi emosi EmoLogus ditujukan untuk mengetahui nilai emosi yang terkadung didalam sebuah kalimat percakapan di dalam bahasa perancis pada anak-anak berusia antara 5-9 tahun. Proses pengolahan kalimat untuk mendapatkan nilai emosi meliputi proses pemisahan kalimat kedalam bentuk kata, pembentukan kata agar sesuai dengan struktur bahasa perancis, proses pemahaman kalimat, dan perhitungan nilai emosi. Nilai emosi di definisikan dalam nilai positif dan negative dengan range -2 sampai +2. Nilai negative mewakili perasaan jijik, ketakutan, kemarahan, dan kesedihan, nillai nol mewakili rasa netral, dan nilai positif mewakili rasa gembira dan kejutan. Penelitian ini ditujukan untuk merancang ulang sistem Emologus untuk menghasilkan sebuah aplikasi baru yang lebih mudah di gunakan oleh pengguna awam, sehingga kedepannya EmoLogus tidak hanya diterapkan untuk robot saja, tapi juga untuk aplikasi-aplikasi komputer, mobile dan lain-lain. Dalam penelitian ini digunakan bahasa pemrograman java untuk desain interface grafis dan bahasa pemrograman λProlog yang digunakan untuk pengembangan Emolgus. Hasil penelitian menunjukan bahwa Java dapat berkomunikasi dengan baik dengan bahasa pemrograman λProlog dan aplikasi yang dihasilkan lebih user mudah digunakan yang mana sebelumnya tidak dapat dilakukan dengan bahasa pemrograman λProlog.

Kata Kunci : *EmoLogus, λProlog, Java, lexiques, parsing, segmentasi, emosi, robot pendamping, pemahaman percakapan*

Rekayasa ulang..., Agus Mulyanto, FT UI, 2012

ABSTRACT

Name : Agus Mulyanto
Department : Electrical Engineering
Title : Reengineering of a Emotion Detection System for Robot Companion who Aimed to Children in the Inpatients Hospitals

EmoLogus emotion detection system is intended to determine the value of emotions which is contained in a conversational sentence in French towards children aged between 5-9 years old. The word processing to get the value of emotion includes the sentence separation process into the form of word, formation of words in the structure of French, process of sentence comprehension and calculating the value of emotional. The value of emotion is defined in the positive and negative values with a range of -2 to +2. Negative value represents the feelings of disgust, fear, anger, and sadness. The value of zero represents a sense of neutral. Positive value represents a sense of joy and surprise. This study is aimed to redesign the system of Emologus to generate a new application which is easier to be used by novice users, so that Emologus is not only applied for robots, but also for computer applications, mobiles and others. This study uses java programming language for graphics and interface design and λProlog which is used for the development Emolgus before. The result of this research shows that Java can communicate well with λProlog programming language and the application which is produced is better than before. It could not be done before with λProlog programming language, but now it can.

Keywords : *EmoLogus, λProlog, Java, lexicons, lemmatization, segmentations, emotion, robot companion, spoken language understanding*

Rekayasa ulang..., Agus Mulyanto, FT UI, 2012

RÉSUMÉ

Prenom /Nom : Agus MULYANTO

Domain : Teknik Elektro

Titre : Réingénierie d'un Système de Détection des Émotions pour un Robot Compagnon Destiné aux Enfants en Hospitalisation Longue

Le système de détection des émotions EmoLogus est destiné à déterminer la valeur des émotions contenues dans une phrase de conversation dans la langue française pour les enfants âgés entre 5-9 ans. Traitement des phrases pour obtenir la valeur d'émotions comprennent la séparation des phrases en mots, la formation de mots pour adapter la structure de la langue française, le processus de compréhension de phrases, et le calcul de la valeur de l'émotion. Valeur émotionnelle est définie dans les valeurs positives et négatives avec une gamme de -2 à +2. Les valeurs négatives représentent les sentiments de dégoût, la peur, la colère et la tristesse, un sentiment de la valeur de zéro représente neutre, et les valeurs positives représentent un sentiment de joie et de surprise. Cette étude visait à réingénierie les Emologus système pour produire une nouvelle application plus facilement utilisé par l'utilisateur novice, de sorte qu'EmoLogus futures ne s'applique pas seulement au robot, mais aussi pour les applications informatiques, l'application mobile et autre. Cette étude a utilisé le langage de programmation Java pour la conception d'interface graphique et langage de programmation λProlog utilisé pour développement Emologus. Les résultats montrent que Java peut bien communiquer avec les langage de programmation λProlog et l'application qui en résulte est la facilité d'utilisation plus facile à utiliser qui n'étaient auparavant pas possible avec le langage de programmation λProlog.

Mots-clefs : *EmoLogus, λProlog, Java, lexiques, lemmatisation, segmentations, émotion, robot compagnon, compréhension de la parole*

Rekayasa ulang..., Agus Mulyanto, FT UI, 2012

TABLE DES MATIÉRES

Rekayasa ulang..., Agus Mulyanto, FT UI, 2012

Rekayasa ulang..., Agus Mulyanto, FT UI, 2012

TABLE DES FIGURES

Rekayasa ulang..., Agus Mulyanto, FT UI, 2012

TABLE DES TABLAUX

Rekayasa ulang..., Agus Mulyanto, FT UI, 2012

Chapitre 1
INTRODUCTION

1.1 Introduction Générale

Le projet ANR Emotirob a eu pour objectif le conception de l'élaboration d'un robot compagnon destiné aux enfants en hospitalisation longue est un système qui doit permettre à ce robot de réagir émotionnellement aux énoncés de l'enfant par des expressions faciales et des sons. Le robot peluche détail destiné à accompagner de jeunes enfants de 5 à 7 ans. Le projet mis en œuvre autour de la détection et la simulation des émotions, se situe à l'interface de la robotique et du dialogue homme-machine. Il pose en particulier plusieurs problèmes scientifiques intéressants :

a. Reconnaître la parole et comprendre le langage d'un enfant qui parle à sa peluche (en l'occurrence, le robot) pour détecter son état émotionnel.
b. Doter le robot d'une capacité de réactions émotionnelles pertinentes, et faire exprimer de manière intelligible ces émotions par le robot,
c. Évaluer l'apport de cette interaction : étudier comment mesurer le réconfort d'un enfant, etc.

Les travaux de stage se concentrent sur le système de détection d'émotions EmoLogus. EmoLogus est un composant qui sert à caractériser l'émotion du locuteur en se basant sur le contenu linguistique de ses messages oraux, et non, comme c'est souvent le cas, en conduisant une analyse acoustique du signal de parole [1]. Il se base sur le principe de compositionnalité des émotions: les mots lexicaux possèdent une valeur émotionnelle fixe définie par une norme psycholinguistique, tandis que les verbes et les adjectifs agissent comme des prédicats, dont le résultat dépend de la valeur émotionnelle de leurs arguments [5]. EmoLogus analyse ainsi la structure de l'énoncé pour identifier l'émotion qu'il porte. Il a été comparé avec un système de base (Baseline) qui ne considère

Rekayasa ulang..., Agus Mulyanto, FT UI, 2012

l'énoncé que comme un sac de mots: on se contente ici de sommer les valences émotionnelles des mots qui le composent.

Précisément les travaux de stage sont la réingénierie logicielle du système « EmoLogus » et la poursuite du travail sur l'adaptation du système de valeurs du robot, avec prise en compte de l'historique des discours.

1.2 Plan

Le premier chapitre présente les méthodes de système de détection des émotions pour un robot compagnon destiné aux enfants en hospitalisation longue, le plan du rapport, la mise en contexte du stage, et la présentation de l'établissement d'accueil de stage.

Nous présentons dans le chapitre 2 des notions importantes sur le robot PARO « Mental Comitment Robot », Robot Compagnon EmotiRob, système de Compréhension de parole et de détection des émotions de la langue. Cette présentation est suivie par le chapitre 3 consacré à l'intégration de la technique de détection des émotions avec EmoLogus à savoir : compréhension de parole (système LOGUS), et détection les valeurs de l'émotion.

Dans le chapitre 4, nous décrivons l'ensemble des travails au cours de la réingénierie Emologus dans l'interface graphique.

Enfin, nous concluons ce stage en montrant dans quelle mesure les objectifs fixés ont été atteints. Nous faisons un bilan du travail effectué et nous présentons les perspectives.

1.3 Contexte

Le projet EmotiRob vise à créer un robot compagnon destiné à des enfants fragilisés (par exemple, des enfants en hospitalisation longue) de 5 à 7 ans. L'objectif est de concevoir un robot-peluche autonome "réactif", susceptible

Rekayasa ulang..., Agus Mulyanto, FT UI, 2012

d'apporter un peu de réconfort à ces enfants. Le robot doit être capable d'exprimer des émotions élémentaires et le robot doit « comprendre » les propos de l'enfant et déterminer son état émotionnel, pour apporter la réponse la plus appropriée.

Ce stage travaille sur EmoLogus qui est la partie du robot EmotiRob en charge de transformer une séquence ou d'un graphe de mots en une représentation sémantique et de détecter les émotions à partir du contenu linguistique des énoncés. Ce stage s'attaque à plusieurs fronts qui s'articulent autour du langage Lambda-Prolog et Java. Lambda-Prolog est utilisé pour le traitement de la compréhension du langage parlée suivant les étapes suivantes: Analyse de la phrase et transformation d'une séquence de mots en une représentation sémantique, puis détection des émotions.

La première tâche demandée au cas de ce stage consiste à rendre le système plus opérationnel et plus convivial. Car les données générées par Emologus difficile à lire et à l'analyse. Données sous la forme d'un texte dans le format de λterme et le forme peu chaotique ou brute. Java est utilisé pour exécuter l'application EmoLogus, afficher les sorties de traitement de EmoLogus dans une interface graphique et gérer les lexiques. L'interface permet par exemple l'affichage des résultats de segmentations, de la structure sémantique issue de l'analyse de l'énoncé et de la valeur émotionnelle calculée par le système.

1.4 Objectifs

Ce stage doit répondre aux besoins du projet EmotiRob. De manière générale, on peut dire qu'il a pour but la détection de l'émotion. Détails du travail durant le stage à savoir :

a. Le premier objectif est la réorganisation du lexique de Logus et la réorganisation de la connaissance sur les valeurs émotionnelles. Pour cela, nous avons commencé par modifier la liste des lexiques de Logus, il faut préciser que le système avait été dans 2 domaines d'applications

Rekayasa ulang..., Agus Mulyanto, FT UI, 2012

différentes à savoir le domaine de renseignement touristique et le domaine des enfants.

b. Le deuxième point de cette étude consiste à créer une interface graphique qui permette :
   - De compiler une application donnée pour un domaine donné. C'est à dire de sélectionner dans un menu les fichiers de lexique associés à un domaine donné (contes, hôtels ou tout autre) et de lancer via l'application graphique, la compilation et la génération d'un exécutable correspond dans ce domaine.
   - Une fois qu'un exécutable est généré, de lancer le système EmoLogus pour un domaine à choisir dans un menu.
   - Pouvoir modifier, ajouter et supprimer des entrées du lexique qui sont stockés dans un fichier au format λ-Prolog.

1.5 Présentation de l'établissement d'accueil

L'IRISA-UBS est un laboratoire de recherche en informatique de l'Université de Bretagne Sud qui développe ses activités dans le domaine de l'informatique diffuse et de « l'intelligence ambiante » en intégrant trois voies complémentaires de recherche :

a. Les systèmes logiciels interactifs multimédia et « intelligents » en tant que support à une « intelligence ambiante ».
b. L'architecture des systèmes logiciels dédiée à la maintenance, au test et à l'évolution dynamique des composants distribués en tant que support à une « informatique ambiante ».
c. Les intergiciels pour les systèmes distribués mobiles et communicants en tant que support à une « informatique ubiquitaire et diffuse ».

Ce stage travaille sur l'équipe RIMH (robotique et l'interaction multi-modale pour le handicap). L'axe de recherche RIMH couvre trois groupes à savoir :

a. Interaction humain-machine

Rekayasa ulang..., Agus Mulyanto, FT UI, 2012

Le groupe interaction humain-machine mène des recherches dans le domaine modélisation de l'interaction, collaboration, multidispositif (MICMAC) qui aide à la communication : l'entrée de texte, l'interaction ubiquitaire, l'interaction dégradée, l'interaction multimodale, et la modélisation de l'interaction sur grandes surfaces.

b. Robotique

Les recherches menées par le groupe robotique se présentent comme suit :

- Génération d'émotions pour robot thérapeutique (Robot assisted therapy)
- Conception de structures robotiques reconfigurables (atomes robotiques)
- Langage de programmation de structures robotiques

c. Langage naturel

Ce stage est plus précisément fait dans ce groupe qui a réalisé la recherche de la modélisation et compréhension du dialogue par approche logique.

Rekayasa ulang..., Agus Mulyanto, FT UI, 2012

Chapitre 2
THEORIQUE

2.1. Robot Compagnon : *Mental Commitment Robots* « PARO »

Les robots compagnon sont conçus pour interagir avec les êtres humains et leur apporter du réconfort. Leur fonction est d'engendrer des émotions, telles que le plaisir et la détente, en tant que robots personnels. Les robots compagnons sont utilisés, par exemple dans le milieu médical (thérapeutique) **(**Anglais **:** *Mental Commitment Robots).* Les robots compagnons de thérapeutiques sont conçus pour fournir 3 types d'effets [1] à savoir :

a. Les effets psychologiques, tels que la relaxation et la motivation,
b. Les effets physiologiques, tels que l'amélioration des signes vitaux,
c. Les effets sociaux tels que l'incitation la communication entre les patients hospitalisés et les soignants.

Le robot PARO est un robot phoque thérapeutique, montré dans la figure 2.1, qui a été développé par l'AIST, l'un des principaux pionniers japonais de l'automatisation industrielle. Son apparence est calquée sur un bébé phoque, et sa surface est recouverte d'une fourrure blanche pure. Choisir une forme animale apporte un effet positif pour les humains (le stress peut être réduit en ayant un animal [12]. Le Robot PARO a été utilisé dans une expérimente de projet EmotiRob pour étudier l'impact d'utilisation les animaux sur l'apaisement psychologique et physiologique de la personne handicapée ou sur leur socialisation.

---

[1] http://paro.jp

Rekayasa ulang..., Agus Mulyanto, FT UI, 2012

Figure 2.1 Robot thérapeutique « PARO »

Paro a cinq types de capteurs [1] : tactiles, de lumière, d'auditions, de température et capteurs posture, avec lequel il peut percevoir les gens et leur environnement. Avec le capteur de lumière, Paro peut reconnaître la lumière et l'obscurité. Il sent s'il est caressé et battu par le capteur tactile, ou s'il est tenu par le capteur posture. Paro peut aussi reconnaître la direction de la voix et des mots tels que son nom, des salutations, et des louanges avec son capteur audio.

Paro peut apprendre à se comporter de la manière que l'utilisateur préfère, et répondre à son nouveau nom. Par exemple, si vous le caressez à chaque fois que vous le touchez, Paro se souviendra de votre action précédente et il essaiera de répéter l'action d'être caressé. Si vous le frappez, Paro se souvient de son action précédente et essaie de ne plus faire cette action. En interaction avec les gens, Paro réagit comme s'il était vivant, bouge la tête, émet des sons, et montre votre comportement préféré. Paro imite aussi la voix d'un bébé.

KAZUYOSHI WADA a mené des recherches en 2007 [16] pour le traitement des patients souffrant de démence clinique à un neurone cortical en utilisant le robot Paro. Les résultats des expériences préliminaires montrent que la thérapie-robot possède un fort potentiel pour améliorer l'état de l'activité cérébrale chez les patients souffrant de démence. Cela est particulièrement vrai pour les patients qui aiment beaucoup PARO.

[1] http: //www.parorobots.com

Rekayasa ulang..., Agus Mulyanto, FT UI, 2012

2.2. Robot Compagnon : Projet ANR EmotiRob

Un nouveau défi dans le domaine de la robotique est la création de robots qui peuvent interagir avec les humains. Le robot doit être capable de s'exprimer en tant qu'n humain tel que triste, heureux, en colère, etc. Le projet ANR EmotiRob est un projet pour créer un robot compagnon peluche destiné à de jeunes enfants de 5 à 7 ans souffrant de divers handicaps, pour les réconforter et les accompagner dans leur vie quotidienne [Sébastien Saint-Aimé, et al. 2009].

Un robot compagnon est équipé de capacités à détecter le sens des messages oraux produits par l'enfant (compréhension de la parole) mais également à détecter les buts communicationnels (actes de dialogue par exemple) qui leur sons associés. Pour mener à bien ce projet, le programme de recherche comprend la collecte des enregistrements de langage d'enfant et la construction d'un corpus permettant des études linguistiques dans les contextes du langage l'enfant.

Les objectifs principaux d'EmotiRob sont de fournir du réconfort aux enfants vulnérables et/ou à ceux subissant une hospitalisation de longue durée avec l'aide d'un robot qui peut être utilisé comme un compagnon affectif et de créer un robot compagnon qui a les capacités suivants [6]:

- Reconnaissance et la compréhension de la langue parlée d'un enfant.
- Interaction émotionnelle entre l'enfant et le robot.
- Interaction cognitive entre l'enfant et le robot.

Ce robot se compose essentiellement de quatre principaux éléments interdépendants dans la figure 2.2, à savoir : (Ce stage s'est concerné par cadres EmoLogus : la compréhension du langage parlée « LOGUS » et la sémantique des émotions)

a. Entrées : Le système de reconnaissance de la parole est fondé sur la transformation de signaux sonores vers le texte. Puis les textes sont analyses par Emologus.

Rekayasa ulang..., Agus Mulyanto, FT UI, 2012

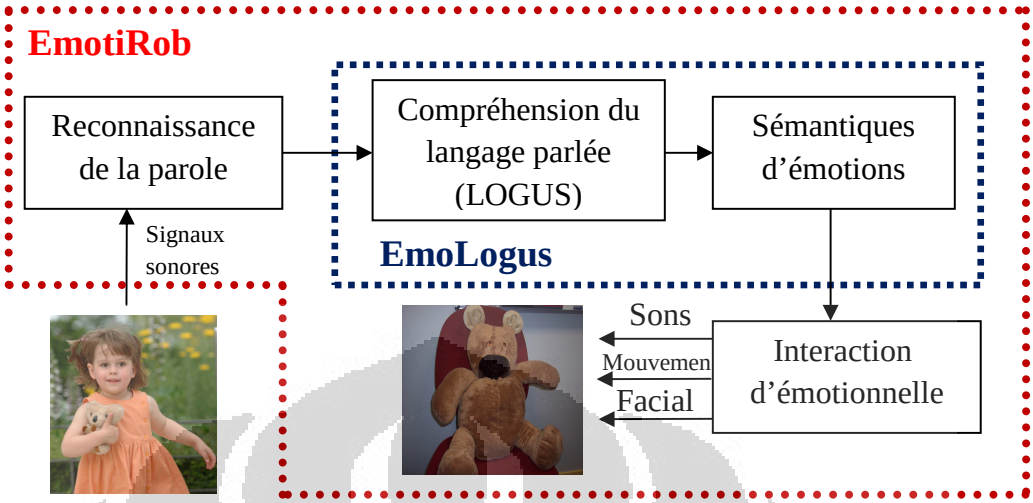


Figure 2.2. Synoptique général du projet

b. EmoLogus se compose de deux systèmes à savoir : le système de compréhension du langage parlé et le système de détection des émotions. La tâche du système de compréhension du langage parlé (LOGUS) est de transformer une séquence ou un graphe de mots en une structure sémantique « formule logique ». Le module sémantique des émotions permet la détection des valeurs émotionnelles des mots de l'énoncé et de sa structure sémantique « formule logique » qui décrit précisément les relations des mots entre eux. L'EmoLogus sera expliqué plus en détail au chapitre 3.

c. Interaction d'émotionnelle : Le module d'interaction émotionnelle est comme le cœur ou le cerveau du système. Il reçoit des informations d'entrée, il traite avec des algorithmes différents et détermine le comportement du robot : les sons, la posture et les états facials (joie, colère, surprise, dégoût, tristesse et peur) qui doit être basé sur le discours [7].

d. Module de sorties : En sortie du modèle robot-compagnon doit être capable de s'exprimer en fonction des caractéristiques matérielles qui le

Rekayasa ulang..., Agus Mulyanto, FT UI, 2012

composent : micro, et moteurs. Le comportement qui découle du module d'interaction émotionnelle sera décomposé en 3 parties principales :

1. Sons « de la voix » : caractérisé par un niveau plus ou moins élevé du signal sonore et du choix du son qui sera produit par le robot. L'interaction étant non verbale dans le cadre de mes travaux, le robot-compagnon devra être capable d'émettre des sons sur le même ton que ceux du robot phoque « Paro ». Ces petits sons « *Sounds into Syllables* », sont des notes de piano associées aux émotions primaires.
2. Posture : caractérisé par la vitesse et le type de mouvement que devra réaliser les différents membres du corps qui composent notre robot, en relation avec le comportement généré.
3. Etat facial : représente les expressions faciales qui doivent être affichée sur le visage du robot. Celles-ci devront par la suite être traduites en émotions primaires, puis en expressions faciales. Les expériences émotives sont composées de plusieurs émotions primaires comme sur le tableau 2.1 [8,9].

Tableau 2.1 Exemple association entre expériences émotionnelles et émotions

Emotion	Expériences émotionnelles
Joie	Amusement, le bonheur, la gaieté, etc.
Colère	Rage, colère, fureur, la colère, l'hostilité, etc.
Surprise	Stupeur, surprise, d'étonnement, etc.
Dégoût	Dégoût, répulsion, de mépris, etc.
Tristesse	Dépression, désespoir, d'impuissance, etc.
Peur	Anxiété, nervosité, tension, etc.

2.3. Système de Compréhension du Langage Naturel

La compréhension du langage naturel dans un système de dialogue homme-machine est un problème complexe. Le procès de traitement est connu comme Le Traitement Automatique du Langage Parlé (TALP). Le TALP vise

Rekayasa ulang..., Agus Mulyanto, FT UI, 2012

l'analyse robuste de la structure linguistique (syntaxe, sémantique, pragmatique) des énoncés de parole spontanée, à partir de transcriptions (séquence ou treillis de mots) obtenues par reconnaissance automatique de la parole. L'architecture globale du système de dialogue de parole décrit dans la figure 2.3 ci-dessous [13]:

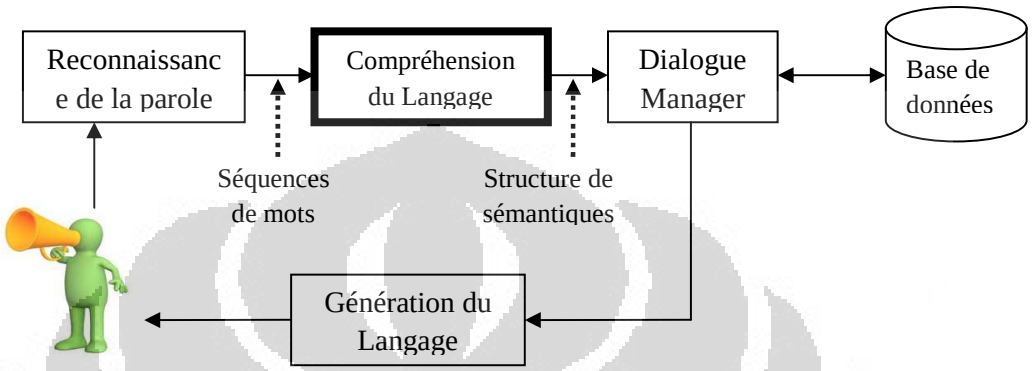


Figure 2.3. L'architecture du système de dialogue de parole

La reconnaissance de la parole est un module pour transformer les signaux des sons en texte. Les problèmes principaux dans cette partie à savoir : présence des erreurs de reconnaissance qui corrompent fortement les transcriptions, et présence de disfluences orales (répétitions, reprises, incises) qui cassent la régularité syntaxique des énoncés. Plusieurs méthodes et algorithmes sont utilisés pour améliorer les performances des systèmes, par exemple Hidden Markov Models et dynamic time warping (DTW).

Jeans-Yves Antoine en 2004 a mis en œuvre une stratégie incrémentale d'analyse robuste (robust parsing ou incremental shallow parsing). L'énoncé est soumis à des traitements successifs qui répondent aux mêmes exigences [2,14] :

a. Analyse superficielle, au sens où la profondeur des représentations élaborées ne s'accroit que légèrement d'un niveau à l'autre : par exemple, l'analyse syntaxique est décomposée en une étape de *chunking* (segmentation en constituants syntaxiques minimaux) suivie par une recherche de rattachement des *chunks*,

Rekayasa ulang..., Agus Mulyanto, FT UI, 2012

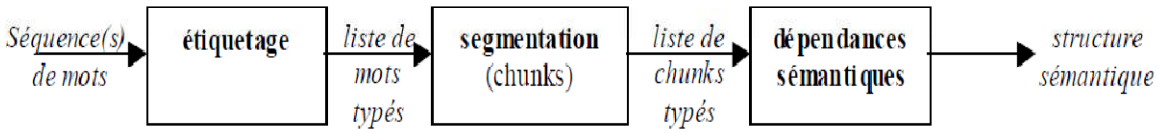


Figure 2.4. L'architecture du système compréhension de la langue [2]

b. Traitements sous-spécifiés et analyse non destructrice : on ne prend de décision qu'en cas de forte confiance. Par exemple, on peut choisir de conserver une ambiguïté qui sera levée par un niveau ultérieur, plutôt que de faire une erreur,
c. Indépendance conceptuelle de chaque niveau : les connaissances manipulées font sens par elles-mêmes,
d. Mise en œuvre à l'aide de techniques d'analyse efficaces sous forme de machines à états finis (par exemple).

L'exemple suivant est une phrase sortie du module de reconnaissance de la parole :

« *il y a euh quels sont les restaurants dans dans le coin* », dans cette phrase on trouve une répétition mots « *dans* ». Module Compréhension du Langage va corriger le phrase en supprimant un mot de deux mots « *dans* ». Les explications plus détaillées seront données dans le chapitre trois.

Les sorties de Module Compréhension du Langage sous la forme de structure sémantique sont traitées par le module Dialogue Manager qui génère une phrase de réponses pour communiquer avec l'homme.

2.4. Détection d'émotions

Une émotion est une réaction psychologique et physique à une situation (expression, faciale et vocale)[1]. La détection des émotions véhiculées par un message oral peut se faire sur un message oral complet, un paragraphe, une phrase ou un tour de parole dans le cas du dialogue oral homme-machine [1].

[1] http://fr.wikipedia.org/wiki/émotion

Rekayasa ulang..., Agus Mulyanto, FT UI, 2012

Les émotions dans un message oral sont classées en trois catégories souvent appelées modalités émotionnelles à savoir : valence positive, négative ou neutre. On peut utiliser une classification en états émotionnels plus détaillée telle que : le neutre, la colère, la joie, le dégout, la peur, la surprise et la tristesse [1,2,3,14]. Deux approches principales retiennent l'attention en matière de caractérisation des émotions [4] : l'approche catégorielle et l'approche dimensionnelle.

L'approche catégorielle est basée sur un ensemble d'émotions dites « basiques », universelles, non réductibles et innées. Ces émotions seraient apparues aux cours de l'évolution et auraient un rôle adaptatif, permettant de réagir à des événements extérieurs, potentiellement dangereux. Selon ce point de vue, on peut remarquer que sept émotions principales se retrouvent chez la plupart des auteurs : outre l'émotion neutre, ce sont la colère, le dégout, la joie, la peur, la surprise et la tristesse.

L'approche dimensionnelle vise à réaliser une catégorisation ordinale décrivant l'état émotionnel dans un espace multidimensionnel. Les dimensions correspondent aux dimensions valence et intensité. Selon ce point de vue, il existerait des états plutôt négatifs et d'autres plutôt positifs, chacun d'eux étant marqué d'un niveau d'intensité élevé ou au contraire d'un niveau d'intensité plus feuille.

Dans le projet EmotiRob « Emologus » la détection des émotions se fait selon l'approche dimensionnelle. L'objectif est étudier l'émotion portée par le contenu linguistique des énoncés. EmoLogus se limite à une caractérisation des émotions en valence (positive « la joie et la surprise », nulle « le neutre » ou négative « le dégout, la peur, la colère, et la tristesse ») associée à un degré d'intensité.

Rekayasa ulang..., Agus Mulyanto, FT UI, 2012

Chapitre 3
EMOLOGUS

EmoLogus est un composant qui sert à caractériser l'émotion du locuteur en se basant sur le contenu linguistique de ces messages oraux, et non, comme c'est souvent le cas, en conduisant une analyse acoustique du signal de parole [1]. Il se base sur le principe de compositionnalité des émotions: les mots lexicaux possèdent une valeur émotionnelle fixe définie par une norme psycholinguistique, tandis que les verbes et les adjectifs agissent comme des prédicats, dont le résultat dépend de la valeur émotionnelle de leurs arguments [5]. EmoLogus analyse ainsi la structure de l'énoncé pour identifier l'émotion qu'il porte.

Cette structure est fournie par le système LOGUS de compréhension de la parole. Si les principes de Logus ont été repris pour construire la représentation sémantique des énoncés, les ressources utilisées pour le fonctionnement ont du être revus. En effet le domaine de test de Logus était le renseignement touristique, à destination d'adultes. Dans le cadre du projet EmotiRob, le public choisi se compose d'enfant dont l'âge peut varier de 5 à 9 ans. Quant à la tâche, si pour l'évaluation du système Logus, il était clairement défini, ici il n'y a pas de tâche clairement visé. Il est évident qu'il parait impossible et irréalisable de faire un système de compréhension tout azimut, cela nécessiterait une connaissance sémantique irréalisable, capable de faire le regroupement de tous les éléments qui composent les énoncés.

3.1 LOGUS : Le système de compréhension du langage parlé

LOGUS est un système de compréhension du langage parlé dans le contexte d'un dialogue Homme-Machine finalisé. L'objectif qui a prévalu pendant le développement de Logus est concilier finesse de l'analyse et une certaine robustesse, nécessaire notamment pour le traitement de la langue oral.

Rekayasa ulang..., Agus Mulyanto, FT UI, 2012

Finesse de l'analyse est l'analyse se voulait « fine » parce qu'elle prétend rendre compte d'une manière aussi précise que possible du sens de l'énoncé [4]. Elle se trouve donc confrontée à une double complexité :

- Pour pouvoir traduire aussi fidèlement que possible les intentions du locuteur, elle devait pouvoir analyser des indices qui permettent de détecter une large palette d'actes de dialogue.
- Elle devait également être capable de reconnaître et de modéliser les objets liés au domaine ainsi que leurs propriétés. Le renseignement touristique faisant déjà apparaitre une certaine complexité de ces objets (« le tarif des chambres doubles ou simples au Caumartin et au Crillon »).

L'analyse se veut « robuste » et donc capable d'analyser des énoncés modifiés par la nature même des énoncés oraux [4]. Elle devait pouvoir résister notamment:

- aux phénomènes relatifs au « travail de formulation » : hésitations, reprises, auto corrections, etc.
- aux phénomènes relatifs à l'interaction : « phatiques » qui émaillent les énoncés oraux: « quoi », « bien », « bon », etc.
- à la souplesse de l'organisation syntaxique ou sémantique des énoncés oraux : organisation en strates successives, mouvements des constituants (dislocations, structures clivées ou pseudo-clivées etc.), changements de parcours syntaxiques et sémantiques (faux-départs, incises, etc.),
- à un nombre non négligeable d'erreurs de reconnaissance. Il s'agit donc de trouver un juste équilibre entre l'obligation de pouvoir ne pas tenir compte de certains éléments (robustesse) et la volonté de traduire aussi précisément que possible ce que le locuteur veut dire (finesse).

La figure 3.1 ci-dessous décrit l'architecture globale du système de compréhension LOGUS.

Rekayasa ulang..., Agus Mulyanto, FT UI, 2012

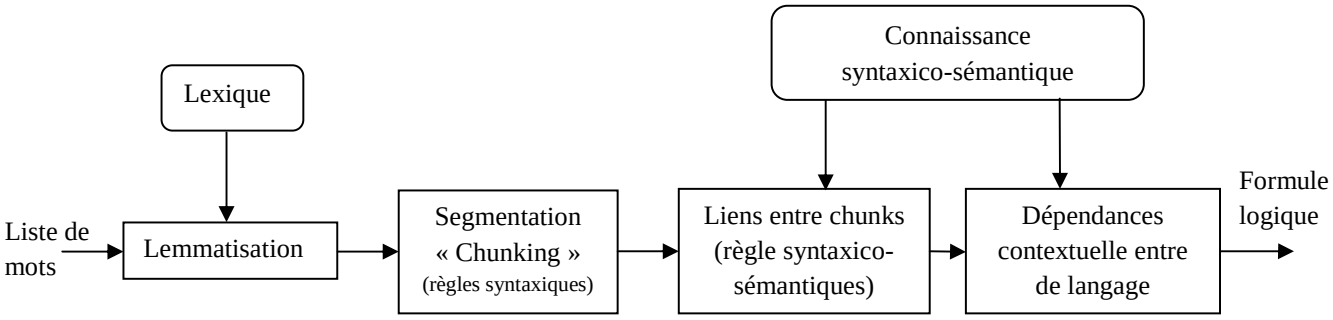


Figure 3.1 L'architecture globale du système de compréhension LOGUS [4]

LOGUS se compose de quatre processus principaux en charge du traitement des données d'entrée « liste de mots » dans une formule logique, ils sont le processus de lemmatisation, le processus de segmentation, et le processus de lier entre chunks et le processus de dépendances contextuelle entre de langage.

A. Lemmatisation

Dans EmoLogus lemmatisation sont les étapes de la formation des mots pour adapter la structure des mots dans un dictionnaire français. La première étape de ce processus consiste à séparer chaque mot d'une phrase, et on relie les formes aux entrées du lexique en basée sur les règles de la grammaire française. Par exemple :

Phrase : **Le chat mange le rat**

La séparation chaque mot d'une phrase:

Phrase :	**Le**	**chat**	**mange**	**le**	**rat**
Pos :	0	1	2	3	4	5

La formation des lexiques : *« le chat » « mange » « le rat »*

Chaque entrée du lexique est définie par la liste de ses définitions (la représentation sémantique) de sorte qu'un lexème peut posséder une ou plusieurs définitions. La représentation sémantique est l'image donnée par le

Rekayasa ulang..., Agus Mulyanto, FT UI, 2012

système du sens de l'énoncé. Elle doit donc être conçue en fonction des besoins du système de dialogue. La représentation sémantique est représentée par trois éléments qui sont appelles « triplets C; R; T » [14]. Les descriptions des triplets sont les suivants :

a) Le premier élément C est appelé catégorie syntaxique. Il joue le rôle d'une étiquette syntaxique. Lors de la segmentation, il intervient en tant que catégorie au sens des grammaires catégorielles. Au niveau de l'étude des dépendances entre chunks, il permet d'imposer des contraintes syntaxiques dans les règles de composition.

Tableau 3.1 Les principales catégories syntaxiques utilisées dans LOGUS [14]

No	Catégorie	Définition
1	(det *nature *nombre)	déterminant : **nature* : défini, possessif, etc. **nombre* : singulier, pluriel
2	(preposition *nature_prep)	préposition : **nature_prep* : ad,in,ab,etc.
3	nomc	nom commun
4	nomp	nom propre
5	(c_nom *categ *determinant)	*chunk* nominal **categ* : catégorie de la tête lexicale
6	(c_nom_p *categ *prepos)	*chunk* nominal prépositionnel
7	(pronom *nature *pers *nombre)	les pronoms pers : 1iere, 2ieme ou 3ieme personne
8	(pre_c_nom *categ)	*chunk* lexical début d'un *chunk* nominal
9	(pre_c_nom_p *categ * prepos)	*idem* précédé d'une préposition
10	(adj_num *ordr)	adjectifs numéraux *ordre : ordre de grandeur
11	adjectif	adjectifs
12	c_adj	chunk adjectival
13	adjectif_pre_pose	adjectifs placés avant le nom qu'ils déterminent
14	adverbe	adverbes
15	(superlatif *nombre *nature_prep)	les superlatifs
16	(comparatif *comp. *categ)	les comparatifs
17	expr	expressions diverses

Rekayasa ulang..., Agus Mulyanto, FT UI, 2012

18	infinitif	infinitif
19	(infinitif_p *categ)	infinitif introduits par préposition
20	(pp *auxi)	participe passé **auxi* : auxiliaire qu'il gouverne
21	(verbe *pers *temps)	verbe conjugué **temps* : passé, présent, futur
22	(gv *pers *temps)	sujet + verbe
23	interjection, conjonction, particule	interjections, conjonctions et particules
24	(categs C1 C2)	catégorie syntaxique résultat de l'association de deux constituants de catégories resp C1 et C2
25	(conj C1 C2 coord)	catégorie syntaxique résultat de la coordination de deux constituants de catégories resp. C1 et C2

Tableau 3.2 Exemples de catégories syntaxiques

No	Exemple	Catégorie Syntaxiques
1	« les »	(det def plur)
2	« dans »	(preposition in)
3	« chat »	nomc
4	« ascension »	nomp
5	« cet après midi »	(c_nom nomc (det dem sing))
6	« vers la gare »	(c_nom_p (c_nom nomc (det def sing)) (preposition ad))
7	« en chambre »	(c_nom_p nomc (preposition ad))
8	« celui »	(pronom indef 3 sing)
9	« un grand »	(pre_c_nom adjectif_pre_pose (det indef sing))
10	« dans un deux »	(pre_c_nom_p (pre_c_nom (adj_num \inf10") (det indef sing)) (preposition in))
11	« pas trop cher »	c_adj
12	« dans le plus cher »	(superlatif sing (preposition in))
13	« moins d'étoiles »	(comparatif moins_que (c_nom_p (preposition de_prep)))
14	« pour aller »	(infinitif_p (preposition pour))
15	« j'ai mangé »	(gv 1 passe)

Rekayasa ulang..., Agus Mulyanto, FT UI, 2012

16	« du deux au trois »	(categs (pre_c_nom_p (pre_c_nom (adj_num \inf10") (det indef sing)) (preposition de)) (pre_c_nom_p (pre_c_nom (adj_num \inf10") (det indef sing)) (preposition a)))
17	« deux ou trois »	(conj (adj_num \inf10") (adj_num \inf10") ou)

b) Le second élément R est appelé rôle sémantique. Il s'agit d'un étiquetage sémantique des éléments qui indique la fonction sémantique qu'ils occupent dans le langage cible : objets, propriété, etc.

Tableau 3.3 Les principaux rôles sémantiques définis dans LOGUS au niveau des lexiques [14]

No	Rôles sémantiques	Définition
1	obj_acte	objets et actions de l'application
2	(prop *R)	Propriétés *R : étiquette de la propriété
3	Interrogation	forme interrogative
	Requete	marque de requête
	demande	marque d'un ordre (location ou réservation)
	Modalité	marque de la donnée d'une information
4	(coordination *C)	mots de coordination *C : nature de la coordination
	Reprise	marque de reprise
	Excuse	marque d'excuse
	Hesitation	marque d'hésitation
	(auxiliaire *N)	les verbes auxiliaires
5	annexe	pas de rôle sémantique précis
	Neglige	absence a priori de rôle sémantique
	r_inconnu	rôle des mots absents du lexique
6	(unite R)	permet d'obtenir un élément de rôle R lorsqu'on l'associe à un nombre
	(preobj R LC)	permet d'obtenir un élément de rôle R lié à un objet qui satisfait des conditions syntaxiques données dans LC
7	(roles R1 R2)	association des rôles R1 et R2

Rekayasa ulang..., Agus Mulyanto, FT UI, 2012

Les chiffres situés dans les colonnes de gauche de ces deux figures renvoient aux numéros des items ci-après :

1. ***obj_act*** est regroupement des les objets ou actions de l'application. Par exemple : *« se paient »*
2. ***(prop R)*** sont les propriétés où l'attribut R correspond aux étiquettes des propriétés. Par exemple : l'étiquette de propriété « *beaucoup* » est relative aux quantités.
3. Certains rôles désignent des éléments qui donnent des indices sur les intentions du locuteur.
   - ***Interrogation*** : ce peuvent être des interrogations marquées par des mots interrogatifs. Par exemple : « *Quels sont ceux qui .. »* est relative aux interrogations.
   - ***Requete*** : Les mots sont relatifs aux requêtes non interrogatives. Par exemple : *« je souhaite »*.
   - ***Demande*** : Les mots sont relatifs aux demandes explicites. Par exemple : *« restez à l'écoute »*.
   - ***Modalité*** : Les mots sont relatifs de la donnée d'une information, d'une obligation, etc. Par exemple : *« il faut »*.
4. Un certain nombre de rôles sont attribués à des éléments de l'énoncé qui permettent de contrôler certaines compositions, tels que les coordinations, les marques de reprise ou d'hésitation. Par exemples :
   - ***coordination C*** *: « pardon » est relatifs à la coordination non.*
   - ***reprise*** *: « je veux dire »*
   - ***excuse*** *: « excusez »*
   - ***hésitation*** *: « et donc »*
   - ***auxiliaire N*** *: « c'est pas rien »*
5. Un certain nombre de rôles servent à indiquer le caractère non essentiel d'un élément. Par exemples :

Rekayasa ulang..., Agus Mulyanto, FT UI, 2012

- ***annexe*** *: « très bien »*
- ***néglige*** *: « par internet »*

6. Les rôles incomplets correspondent au rôle attribué à un élément qui, associé avec un autre, permettra d'obtenir un rôle donné. Leur utilité est liée au caractère minimaliste des chunks : ils sont destinés à faciliter l'association entre certains mots des lexiques. Par exemple, l'expression « *une heure* » correspond à deux chunks minimaux. L'une des définitions associée au chunk « *heure »* a pour rôle sémantique *(unite (prop horaire)).* Ce rôle indique une association prioritaire de ce chunk avec un chunk de rôle sémantique *(prop entier)* pour obtenir un constituant de rôle *(prop horaire).*
7. Association de rôles : l'application d'un acte de langage à une chaine d'objets, la coordination de deux propriétés correspondent à l'association de deux constituants de rôles différents. le constructeur « *roles »* permet d'en rendre compte. **Définition :** Si R_1 et R_2 sont des rôles sémantiques, alors « *(roles* R_1 R_2 *) »* est un rôle sémantique. Par exemple **:** le rôle sémantique de l'expression « *quels sont les chats* » est *(roles interrogation obj_acte*).

c) Le troisième élément T est la traduction sémantique, traduction du constituant dans la forme l'un λ-terme. Si un constituant contient des éléments lexicaux au sens défini précédemment, sa *traduction sémantique* est sa traduction dans la forme d'un λ-terme. Les autres constituants (marques de coordinations, etc.) et les mots hors-vocabulaire se voient attribuer la traduction sémantique générique de neutre. Lorsque deux éléments de l'énoncé sont assemblés, la traduction sémantique résultante s'obtient essentiellement à partir des traductions sémantiques des deux éléments concernés mais peut également faire intervenir leurs rôles sémantiques.

Rekayasa ulang..., Agus Mulyanto, FT UI, 2012

L'un des obstacles majeur est l'ambiguïté de la langue naturelle, présente à tout niveau: sémantique, syntaxique et pragmatique. Par conséquent, un entrée ambiguë a plus d'une définition. Par exemple dans le phrase ci-dessus est le lexique *« mange »* qui a deux catégories syntaxique : la forme du verbe pour la première personne et le verbe pour une troisième personne.

B. Segmentation « Chunking »

La segmentation en *chunks* permet d'associer des mots juxtaposés sur la base de règles essentiellement syntaxiques. L'objectif de la segmentation est de diminuer d'erreurs de structures syntaxiques qui provoquera des erreurs dans la compréhension d'une phrase.

Le formalisme des grammaires catégorielles de type AB est utilise dans EmoLogus où les règles constituant en d'un triplet < C; R; T >. Deux règles dérivées de celles des [14] sont appliquées aux étiquettes syntaxiques :

- (A ; A\B) → B : composition d'un élément de catégorie *A* avec un élément de catégorie fractionnaire A\B situé à sa droite pour former un élément de catégorie *B*.
- (B/A ; A) → B : composition d'un élément de catégorie fractionnaire *B/A* avec un élément de catégorie A situé à sa droite pour former un élément de catégorie B.

L'orientation de la fonction, « / » ou « \ », s'avère un élément décisif, puisqu'en indiquant l'ordre de dépendance des mots, elle précise tout simplement l'ordre d'apparition des mots dans la phrase. La notation A/A (lue « A sur A »), qui permet de représenter le déterminant, indique ainsi que le nom doit se trouver à la droite de l'article. En fonction de cette représentation des catégories dérivées, des règles syntaxiques vont être créées qui auront pour but de simplifier entre elles les catégories ayant été affectées à chaque expression de la suite de mots que l'on évalue.

Rekayasa ulang..., Agus Mulyanto, FT UI, 2012

La représentation ci-dessus est appliqué en forme le λ-terme qui détermine la partie sémantique du couple résultat s'obtient par composition des λ-termes correspondants. Par exemple, le λ-terme correspondant au chunk adjectival « *pas trop vite* » ne s'obtient comme composition des λ-termes respectivement associés aux trois mots qui le composent:

Lexème	catégorie syntaxique	λ-terme
« pas »	adj/adj	λx.(pas x)
« trop »	adj/adj	λx.x
« vite »	adj	vite
« pas trop vite »	adj	(λx.(pas x) (λx.(x vite))) ≡ β (pas vite))

Cette composition des lamda-termes est réalisée directement par le langage lambda-Prolog avec lequel est développé le système EmoLogus (voir annexe A). Ainsi, l'implémentation des règles des grammaires catégorielles est immédiate avec ce langage.

Voici un exemple de représentation sémantique de phrase « *le chat mange le rat* » se monter ci-dessous :

« le chat » : (c_nom nomc (det def sing)) animals (objprop (etiq chat) (propriete lexGenre (genreNom masc))))

Catégorie syntaxique (C) : (c_nom nomc (det def sing))
Rôle sémantique (R) : animals
Traduction sémantique (T) : (objprop (etiq chat) (propriete lexGenre (genreNom masc))))

« mange » : (verbe t 1 0) action (etiq manger)), (verbe t 3 0) action (etiq manger)

Catégorie syntaxique (C) : (verbe t 1 0) et (verbe t 3 0)
Rôle sémantique (R) : action
Traduction sémantique (T) : (etiq manger)

« le rat » : (c_nom nomc (det def sing)) animals (objprop (etiq rat) (propriete lexGenre (genreNom masc)))

Rekayasa ulang..., Agus Mulyanto, FT UI, 2012

Catégorie syntaxique (C)	: (c_nom nomc (det def sing))
Rôle sémantique (R)	: animals
Traduction sémantique (T)	: (objprop (etiq rat) (propriete lexGenre (genreNom masc)))

C. Liens entre chunks

Le troisième traitement consiste à lier entre eux les constituants sémantiquement interprétés pour obtenir la formule logique finale. Des prédicats logiques décrivent les liens possibles entre les objets et les propriétés. Ils définissent ainsi une connaissance sémantique spécifique du domaine. Les règles utilisées sont exclusivement sémantiques : la stratégie définie dans Logus consiste à fabriquer des « chaînes d'objets » maximales, c'est-à-dire qui intègrent le plus grand nombre possible des objets et propriétés présents dans l'énoncé, en utilisant les liens décrits dans la connaissance sémantique [3]. La détermination de l'acte de langage (ou tout au moins des indices linguistiques présents dans l'énoncé sur la nature de cet acte) s'obtiennent en utilisant une relation d'ordre sur les mots-questions et mots-requêtes.

D. Interprétation contextuelle

L'interprétation contextuelle vise à désambigüiser des sémantiques, réduire les erreurs de l'analyse d'une phrase elliptique ou incomplète durant les dialogues et puis pouvoir compléter des « fragments sémantiques » incomplets [3]. Le principe de résolution repose sur des propriétés syntaxiques et sémantiques. Logus fait donc le choix ici de combiner des indices syntaxiques et sémantiques.

La première étape de ce traitement est de reconnaître des références. Les références sont identifiables soit dans la représentation sémantique elle-même (dans l'exemple, les expressions *« y aller »* et *« d'ici »* ont produit dans la représentation sémantique des objets *objet_contexte* et *lieu_contexte*), soit enfin dans la catégorie syntaxique (dans une expression telle que *« cet hôtel »* , c'est l'adjectif démonstratif *« cet »* qui indique qu'il y a référence à un objet du contexte).

Rekayasa ulang..., Agus Mulyanto, FT UI, 2012

Une fois repérés, il faut encore pouvoir remplacer ces éléments inconnus par le véritable objet. Pour compléter les fragments manquants, il est alors possible d'utiliser des éléments du contexte, rencontrés au cours du dialogue, si la connaissance sémantique le rend possible. La résolution consiste à chercher dans le contexte quels objets peuvent sémantiquement correspondre à ces références. La connaissance syntaxico-sémantique est utilisée une fois de plus, par des règles de rattachement entre les différents objets.

3.2. Détection des émotions dans EmoLogus

L'humain peut détecter de manière assez fiable l'émotion portée par un énoncé en ne considérant que son contenu linguistique. Pour automatiser cette compétence, des connaissances doivent être données au système. EmoLogus caractérise l'émotion du locuteur en se basant sur le contenu linguistique des messages oraux, et non, comme c'est souvent le cas, en conduisant une analyse acoustique du signal de parole. Il se base sur le principe de compositionnalité des émotions: la norme émotionnelle et les lexèmes prédicatifs (la définition de prédicats) [5].

La norme émotionnelle utilisée dans EmoLogus est une valeur émotionnelle qui est associée à des mots du lexique chez les enfants. Les mots lexicaux possèdent une valeur émotionnelle fixe définie par une norme psycholinguistique. EmoLogus analyse la structure de l'énoncé pour identifier l'émotion, calculer la valence émotionnelle des valences lexicales de ses termes et puis classifier d'une valence émotionnelle dans les valeurs positive/négative sur une échelle de centrée de 5 valeurs variant de -2 (émotion très négative) à +2 (très positif) et à associée à un dessin représentant un visage souriant, triste ou neutre [5].

Les mots non prédicatifs du vocabulaire possèdent une valeur émotionnelle intrinsèque (valence), tandis que les mots prédicatifs « *les lexèmes prédicatifs* » (essentiellement les verbes ou les adjectifs) sont associés à des fonctions qui donnent une émotion à partir des valences émotionnelles

Rekayasa ulang..., Agus Mulyanto, FT UI, 2012

des arguments du prédicat [5]. Pour illustrer ce mécanisme, le suivant est un exemple ce que j'ai pris depuis le papier de Le Tallec à 2010. Considérons la phrase : *«le gentil cochon n'a pas d'amis»*, après le processus segmentations, cette phrase a quatre lexicaux « gentil, cochon, avoir, et amis » et la formule logique qui la représente.

- Gentil → E : X → X + 1
- Cochon → E = 0
- Avoir « Emotion dépendante des deux argumentes » → E : (X, Y) → X*Y
- Amis → E = 0

Le calcul commence par la prise en compte de la valence émotionnelle des mots cochon (E=0) et amis (E=1), qui ne sont que de simples arguments de la formule. L'adjectif petit et le verbe avoir fonctionnent comme des prédicats. Gentil ayant comme définition E = X → X +1, le groupe nominal petit cochon va avoir pour valeur E =1. L'application successive de ces prédicats nous permet d'arriver au résultat final pour l'énoncé E= -1. On trouve dans notre lexique une dizaine de fonctions émotionnelles, comme celles présentées en exemple ci dessous :

- Décalage positif E : x → x+1 (exemple : mignon)
- Décalage négatif E : x → x-1 (exemple : énervé)
- Emotion dépendante des deux arguments E : (x,y) → x * y (exemple : perdre)
- Emotion inverse à l'objet E : (x,y) → -y (exemple : casser)

Le principe de compositionnalité sur lequel repose le système est assez simple. Il nécessite toutefois une compréhension robuste de la parole spontanée pour fournir la structure prédicative correcte, et une base lexicale émotionnelle propre sur laquelle se basera le calcul des émotions.

Rekayasa ulang..., Agus Mulyanto, FT UI, 2012

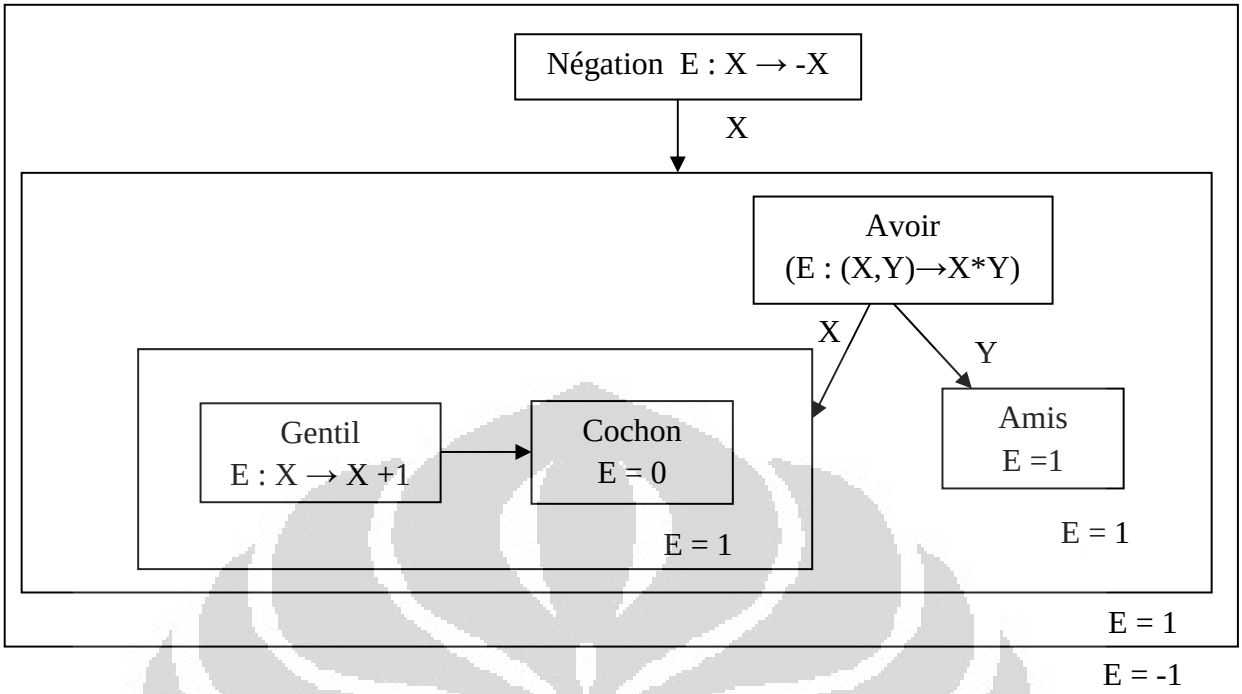


Figure 3.2. Le principe de calcul des émotions dans Emologus

Rekayasa ulang..., Agus Mulyanto, FT UI, 2012

Chapitre 4
IMPLEMENTATION

4.1. Analyse des systèmes existants

Le système de détection des émotions « EmoLogus » pour un robot compagnon destiné aux enfants en hospitalisation longue a été développé en utilisant un langage de programmation λProlog sur le système d'exploitation Linux. Le système Emologus se compose de six modules principaux à savoir : le module d'interface des utilisateur, le module lemmatisation, le module segmentation, le module de données des lexiques, le module de système des compréhensions, et le module de calcul de valeur émotionnelles, telles que celles exposées dans la figure 4.1.

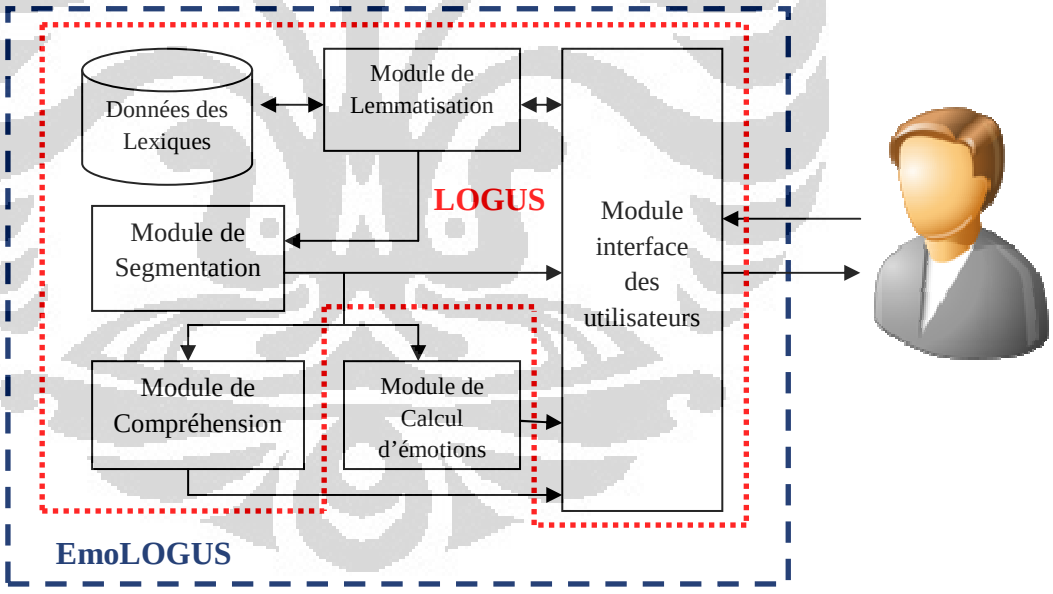


Figure 4.1 La structure des systèmes existants d'Emologus

1. Module d'interface des utilisateurs

   Ce module sert à connecter le système à l'utilisateur. Les capacités détenues par ce module sont de recevoir les données saisies dans le formulaire de texte. Ces données doivent ensuite être traitées par le

Rekayasa ulang..., Agus Mulyanto, FT UI, 2012

module de lemmatisation (en utilisant les données des lexiques), le module de segmentation, le module de compréhensions, et le module de calcul de la valeur émotionnelle. A la fin, le module interface affiche les résultats du traitement à l'utilisateur.

2. Module de lemmatisation

   Ce module sert à séparer la phrase en mots, comme cela a été décrit dans la page 16 et consulter les définitions dans la module de données des lexiques, puis les mots et leur définitions, sont traitées par le module de segmentation.

3. Module de données des lexiques

   Module de données des lexiques est un module dans lequel il ya la liste des mots (ou groupes des mots) lexiques avec leurs définitions qui sont constituées d'un triplet « *catégorie syntaxique (C), rôle sémantique (R), et traduction sémantique (T)*». Il pourrait également être dit que ce module est une base de données du système Emologus. Le lexique compte acttuaellement 6077 entrées en comptant celles des deux domaines d'ontologie, à savoir le domaine des enfants et le domaine de renseignement touristique.

4. Module de segmentation « *Chunking* »

   Comme déjà décrit dans la page 22, la segmentation est utilisée pour associer des mots juxtaposés sur la base de règles essentiellement syntaxiques. Ce module contient un algorithme pour optimiser cette segmentation, afin de regrouper le plus de mots possible. Les résultats de ce processus de segmentation seront utilisés pour le processus de compréhension de phrases et calcul de la valeur d'émotion.

   .

5. Module de Compréhension

   Le module de compréhension contient les commandes pour analyser la structure sémantique d'une phrase ou d'un énonce dans conversation. Le module est divisé en deux sous-modules à savoir : le module de

Rekayasa ulang..., Agus Mulyanto, FT UI, 2012

compréhension des phrases dans un contexte et le module de compréhension d'une phrase hors contexte. Et le module de compréhension hors contexte est un système dont le but de comprendre une seule phrase. Le module de compréhension en contexte est un système de compréhension des phrases d'un énoncé dans laquelle la première phrase et les phrases suivantes ont une relation dans un contexte spécifique.

6. Module de calcul d'émotions

   Ce module contient un algorithme qui sert à traiter les données des résultats du module de compréhension en calculant la valeur des émotions d'une phrase, tel que décrit surla page 19. Le processus de calcul de la valeur émotionnelle sera fait par trois moyens à savoir : le calcul des émotions de base "Base Line", le calcul des émotions de prédicatif simple et le calcul des émotions de prédicatif. La sortie de ce module est la valeur de l'émotion est mis en œuvre avec une valeur numérique de -2 à +2.

4.2. Analyse du système d'exigence

Ainsi que mentionné précédemment, Emologus a été développé en utilisant le langage de programmation λProlog avec des sorties données sont sous la forme d'un λ-terme. Avant le travail effectué au cours de mon stage, es problèmes rencontrés étaient les suivants :

a. Difficulté à comprendre la sortie du système : la lecture d'une lambda-terme n'est pas forcément facile.
b. Utilisation de l'application peu ergonomique : il faut ouvrir la console de Linux lorsqu'on utilise d'application, parce que le langage de e programmation λProlog n'a pas les outils nécessaires pour créer une interface graphique.

Rekayasa ulang..., Agus Mulyanto, FT UI, 2012

c. Difficulté à analyser les erreurs ou l'absence d'une entrée dans le elxique lexique parce qu'il faut ouvrir un module de données des lexiques où les données ont atteint plus que 8000 lignes.
d. Difficulté de distinguer les entrées du lexique utilisées dans le domaine des enfants et le domaine du renseignement touristique, alors que ces deux domaines contiennent l'un et l'autre des vocabulaires relativement spécifiques.

Par conséquent nous avions besoin de revoir l'ingénierie de l'application Emologus pour résoudre ces problèmes, en faisant une application avec une interface graphique, tout en conservant l'application initiale codée en λProlog.

4.3. La proposition

Pour répondre à tous les besoins du système ci-dessus, dans ce stage nous proposons de créer des applications avec une interface graphique en utilisant le langage de programmation Java et de faire Emologus comme un cadre d'application tel que celui représenté sur la figure 8. Voici les propositions que nous fournissons au processus réingénierie d'Emologus.

a. Création d'une application capable de servir de médiateur entre l'utilisateur et Emologus avec un écran convivial.
b. Création d'une application qui peut recevoir une phrase de l'utilisateur à traiter par Emologus et capable d'exécuter des applications Emologus (fichier exécutable d'Emologus) et qui ensuite affiche sa sortie dans un format graphique en forme d'arbre, facilement compréhensible par l'utilisateur ordinaire.
c. Création d'une application capable de gérer (ajouter, modifier et supprimer) les entrées du lexiques qui sont stockées dans la fichier des données des lexiques et puis capable de recompiler l'application Emologus avec ces entrées du lexique.

Rekayasa ulang..., Agus Mulyanto, FT UI, 2012

d. Séparation des lexiques dans deux domaines d'ontologie à savoir de domaine les enfants et le domaine du renseignement touristique.

4.4. Mise en Œuvre

4.4.1. Interface Graphiques

Pour mettre en œuvre la proposition décrite ci-dessus, nous avons conçu le système que nous décrivons sur la figure 4.2 pour développer une interface graphique.

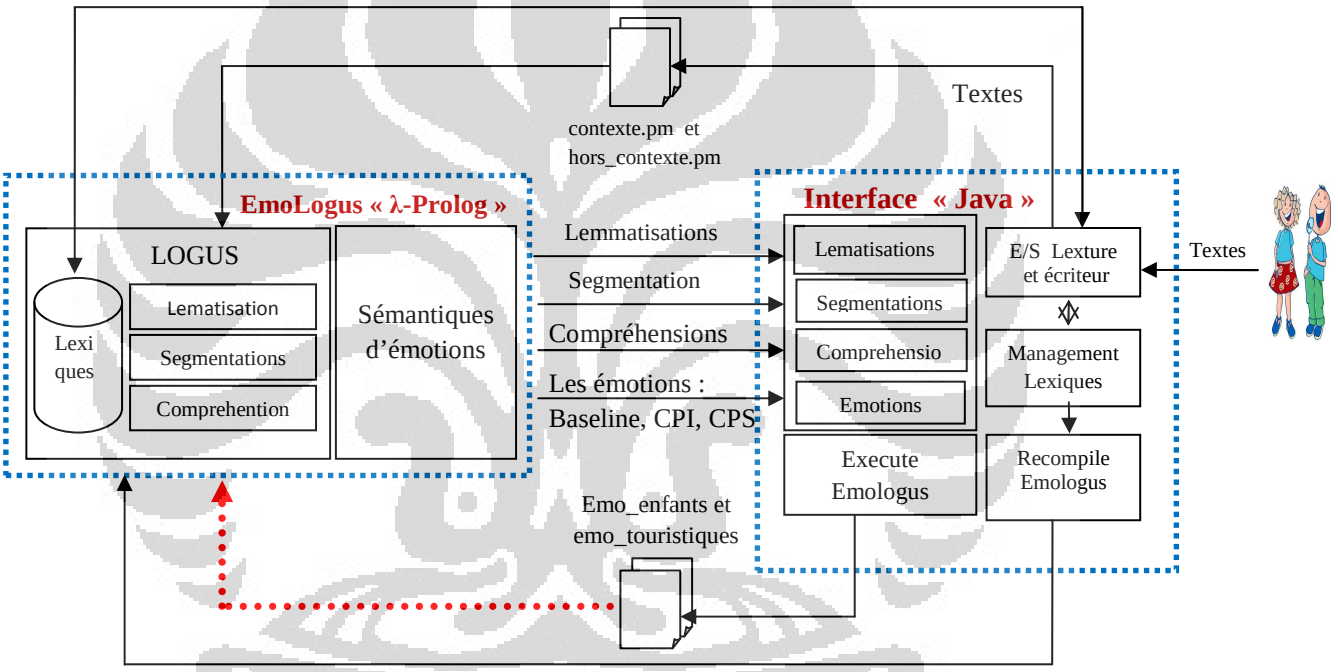


Figure 4.2 La conception du nouveau système d'Emologus

L'interface graphique pour l'application comme le montre la figure 4.2 se compose essentiellement de huit modules principaux, à savoir:

1. Module E/S pour les données d'entrée

   Le module E/S a la tâche d'écrire et de lire des données stockées dans le fichier contexte.pm, hors_contexte.pm et le module de données de lexiques d'Emologus. Le fichier Contexte.pm est le fichier utilisé pour stocker les phrases de conversation qui sont entrées dans l'ordre par l'utilisateur via une interface graphique de

Rekayasa ulang..., Agus Mulyanto, FT UI, 2012

lecture et traitées par Emologus afin d'obtenir la formule logique qui correspond à leur structure sémantique. Le fichier Hors_contexte.pm est un fichier qui a la même fonction que le fichier context.pm, mais ce fichier est utilisé pour stocker des données de conversation en dehors du contexte (une seule phrase).

2. Module de traitement des données lemmatisation

Ce module est utilisé pour traiter des données de sortie du processus lemmatisation de l'Emologus. Les données de sortie de l'Emologus sont en forme des textes comme suivant :

"pos="0"le""/""pos="1"chat""/""pos="2"mange""/""pos="3"le""/""pos="4"rat""/"

"Lemmatisation = " [(pos1 0 2 [(eql (c_nom nomc (det def sing)) animals (objprop (etiq chat) (propriete lexGenre (genreNom masc))))]), (pos1 2 3 [(eql (verbe t 1 0) action (etiq manger)), (eql (verbe t 3 0) action (etiq manger))]), (pos1 3 5 [(eql (c_nom nomc (det def sing)) animals (objprop (etiq rat) (propriete lexGenre (genreNom masc))))])]

Les sorties de ce processus contiennent deux types de deux données à savoir les données positions des mots après le processus de séparation tel que décrit à la page 11 et les données des définitions de chaque lemme, sous la forme d'un triplet « *Catégorie syntaxique (C), Rôle sémantique (R), et Traduction sémantique (T)* » tel que décrit à la page 11. Ces définitions seront utilisées dans le processus de segmentation, la compréhension et le calcul de l'émotion d'une phrase.

A partir de l'exemple ci-dessus peut être expliqué comme suivant :

Position
- 0 à 1 → « le »
- 1 à 2 → « chat »
- 2 à 3 → « mange »
- 3 à 4 → « le »
- 4 à 5 → « rat »

Rekayasa ulang..., Agus Mulyanto, FT UI, 2012

Définitions pos1 0 2 = « le chat »

C = (c_nom nomc (det def sing))

R = animals

T = « chat » avec propriété Genre = Masculine

pos1 2 3 = « mange »

C = (verbe t 1 0)

R = action

T = « manger »

pos1 3 5 = « le rat »

C = (c_nom nomc (det def sing))

R = animals

T = « rat » avec propriété Genre = Masculine

Les données générées à partir de ce module sont les informations de position de chaque mot.

3. Module de traitement des données segmentations

Ce module est utilisé pour traiter les sorties de données du processus segmentations dans l'Emologus et on fait afficher dans une interface graphique en la structure arborescente comme sur monter à la figure 14. Les données de sortie de l' Emologus sont en forme des textes comme suivant :

"segmentations"[(pos2 0 1 [] [(sc_eql 0 (eql (c_nom nomc (det def sing)) animals (obj chat [(propriete quantite L1@(entier 1)), (propriete contexte (qual_ref def L1)), (propriete lexGenre (genreNom masc))]))])), (pos2 1 2 [] [(sc_eql 0 (eql (verbe t 3 0) action (obj manger []))])), (pos2 2 3 [] [(sc_eql 0 (eql (c_nom nomc (det def sing)) animals (obj rat [(propriete quantite L2@(entier 1)),

Rekayasa ulang..., Agus Mulyanto, FT UI, 2012

(propriete contexte (qual_ref def L2)), (propriete lexGenre (genreNom masc))]))))])]

Base sur des données ci-dessus, on ne sait pas les mots qui sont utilisé sur la position de 0 à 1. Alors, il faut utiliser les données qui sont produit par le module de lemmatisation où la position de 0 à 1 du processus lemmatisation remplacer la position de 0 à 2 du processus de segmentations.

Définitions pos2 0 1 = pos1 0 2 = « le chat »
C = (c_nom nomc (det def sing))
R = animals
T = « chat » avec propriété quantité = « *1* », contexte = « *def L1* » (ce contexte sera utilisé dans la processus de compréhension et le calcul de la valeur de l'émotion), genre = « *masculin* »

pos2 1 2 = pos1 2 3 = « mange »
C = (verbe t 3 0)
R = action
T = « manger »

pos2 2 3 = pos1 3 5 = « le rat »
C = (c_nom nomc (det def sing))
R = animals
T = « rat » avec propriété quantité = « *1* », contexte = « *def L2* » (ce contexte sera utilisé dans le processus de compréhension et le calcul de la valeur de l'émotion), genre = « *masculin* »

Rekayasa ulang..., Agus Mulyanto, FT UI, 2012

La figure 4.3 ci-dessous monte la forme d'arborescence de processus segmentations et lemmatisations.

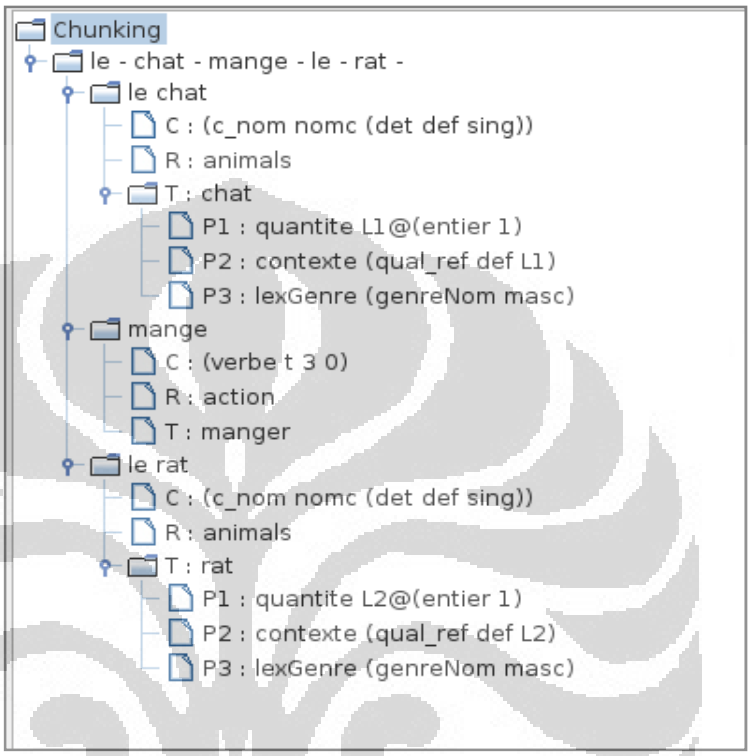


Figure 4.3 L'arborescence de processus de segmentation et lemmatisation.

4. Module de traitement des données : compréhension

Tout comme le module de lemmatisation et le module de segmentation, ce module est utilisé pour traiter les sorties de données du processus compréhension dans l'Emologus. Il y a deux modules de compréhension dans Emologus à savoir le module de compréhension dans le contexte et le module de compréhension hors de contexte. Voici deux phrases pour donner un exemple des données de sortie du module de compréhension en contexte du système Emologus :

Rekayasa ulang..., Agus Mulyanto, FT UI, 2012

Phrase 1 = le loup voit l'agneau

Phrase 2 = il le mange

Ci-dessous les données de sortie Emologus pour comprendre la première phrase :

Compréhension hors contexte :

Formule Logique = [(regles_lchunk [["sujetverbe"], ["verbeeobj"]] [(pos2 0 3 [] [(sc_eql 0 (eql (gv t 3 0) action (obj voir [(propriete cod (obj agneau [(propriete quantite (entier 1)), (propriete lexGenre (genreNom masc))])), propriete sujet (obj loup [(propriete quantite (entier 1)), (propriete lexGenre (genreNom masc))]))]))]))])]

Compréhension dans contexte :

Formule logique en Contexte =[(pos2 0 3 [] [(sc_eql 0 (eql (gv t 3 0) action (obj voir [(propriete cod (obj agneau [(propriete quantite (entier 1)), (propriete lexGenre (genreNom masc))])), (propriete sujet (obj loup [(propriete quantite(entier 1)), (propriete lexGenre (genreNom masc))]))]))]))]

Ci-dessous les données de sortie Emologus pour comprendre la première phrase :

Rekayasa ulang..., Agus Mulyanto, FT UI, 2012

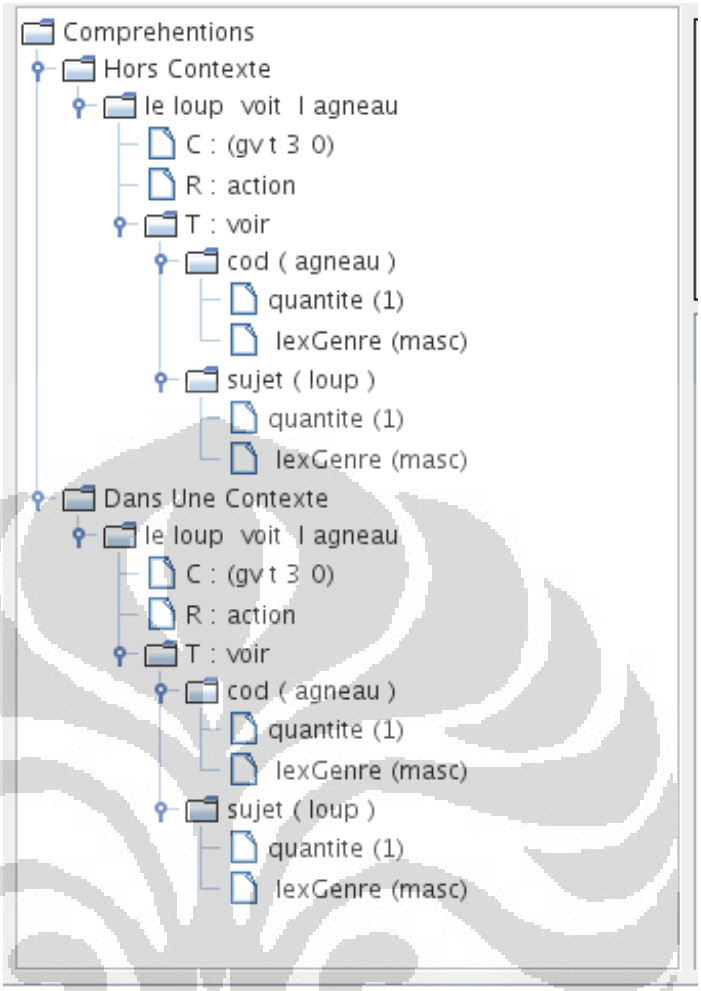


Figure 4.4 L'arborescence de processus de compréhensions de phrase « *le loup voir l'agneau* »

Compréhension hors contexte :

Formule Logique = [(regles_lchunk [["sujetverbeContexte"]] [(pos2 0 2 [] [(sc_eql 0 (eql (gv t 3 0) action (obj manger [(propriete sujet (obj X\ (etiq_contexte_emo 3 masc sing X) [])), (propriete cod (obj X\ (etiq_contexte_emo 3 masc sing X) [])]))))])])]

Compréhension dans contexte :

Rekayasa ulang..., Agus Mulyanto, FT UI, 2012

Formule logique en Contexte = [(pos2 0 2 [] [(sc_eql 0 (eql (gv t 3 0) action (obj manger [(propriete cod (obj agneau [(propriete quantite (entier 1)), (propriete lexGenre (genreNom masc))])), (propriete sujet (obj loup [(propriete quantite (entier 1)), (propriete lexGenre (genreNom masc))]))]))])]]

La figure 4.5 ci-dessous monte la forme d'arborescence de processus compréhensions comprendre la deuxième phrase :

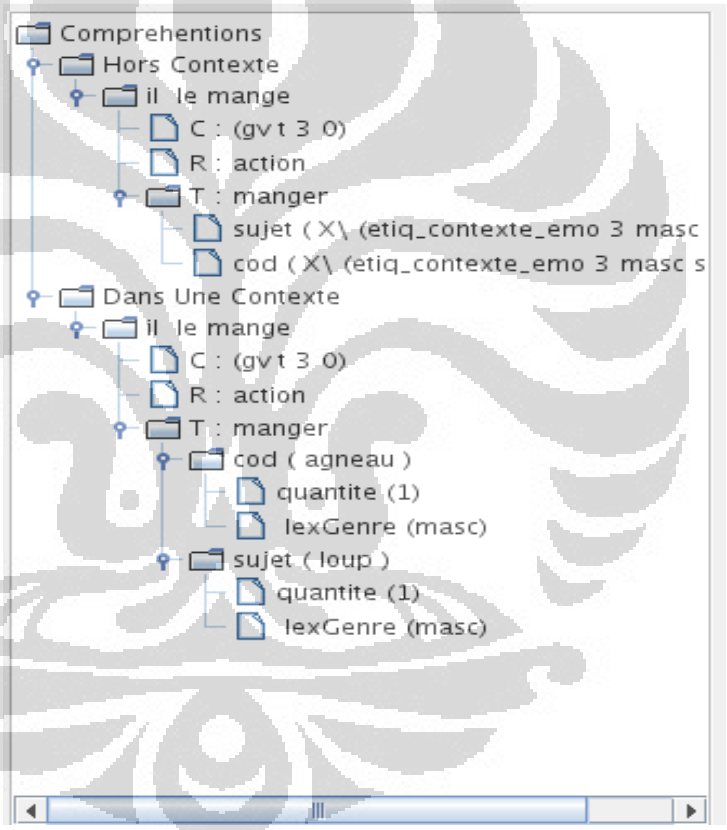


Figure 4.5 L'arborescence de processus de compréhensions de phrase « *il le mange* »

5. Module de traitement des données émotion

Tout comme le module précédent, ce module sert à calculer d la valeur de l'émotion d'un énoncé en sortie de la compréhension. Ce

Rekayasa ulang..., Agus Mulyanto, FT UI, 2012

module permet d'afficher les valeurs émotionnelles par trois méthodes de calcul des émotions qui existent dans Emologus avec des valeurs comprises entre -2 à 2.

6. Module d'exécuter / lancer les applications Emologus

   Comme cela a été discuté dans les chapitres précédents, Emologus a été développé avec le langage de programmation λProlog, bien que l'interface graphique soit créée avec le langage de programmation Java. Les deux sont des langages de programmation qui sont très différents. Ainsi, lorsque nous allons intégrer les deux, on a besoin d'opérer d'une manière spéciale. L'idée intégrer entre les deux es d'exécuter les fichiers exécutables Emologus dans les interfaces graphiques. Java a la capacité d'exécuter des commandes de base du système d'exploitation Linux qui est habituellement exécuté sur la ligne de commande linux. Par exemple java a la capacité d'exécuter une application et de prendre la sortie de données, modifier (voir, lecture et écriture) le contenu du fichier, etc. Donc, ce module contient les commandes de Java qui sont utilisées pour exécuter un fichier exécutable d'Emologus « *emo_enfants* et *emo_tourist* » et prendre les données produites. Voici les commandes utilisées pour exécuter le fichier exécutable et prendre les données:

```
Process Findspace = Runtime.getRuntime().exec
                    (“emo_enfants”);
BufferedReader Resultset = new
     BufferedReader(new InputStreamReader
     (Findspace.getInputStream()));
```

   Le mot « *emo_enfants* » est le nom de fichier exécutable de l'application Emologus, ce fichier est le résultat du processus recompiler par le module « recompiler Emologus ».

7. Module pour gérer le lexique

Rekayasa ulang..., Agus Mulyanto, FT UI, 2012

Ce module est utilisé pour manipuler (ajouter, modifier et supprimer) des données de lexique qui sont stockées dans le fichier principale d'Emologus à savoir *lex_emotirobenfants.pm* et *lex_emotirobtourist.pm*. Pour modifier ce fichier, ce module fonctionne avec le module E/S pour lire et écrire dans le fichier "lex_emotirob.pm".

8. Modules pour recompiler Emologus

Un ensemble des données des lexiques sont stockées dans les fichier lex_emotirobenfants.pm et lex_emotirobtourist.pm en format λterme où toutes les données du lexiques sont définies par l'appellation des variables dans les langages de programmation λProlog qui utilisent le commande « *type* » (voir l'annexe A). Il pourrait également être dit que les lexiques incluent dans les codes le programme de l'Emologus, alors quand nous voulons apporter des modifications aux données des lexiques, nous devons modifier le programme principal (les codes λProlog) d'Emologus. Lorsqu'on va mettre en œuvre les modifications apportées au lexique des données, nous avons besoin de recompiler le programme λProlog pour obtenir de nouvelles applications (nouveau fichier exécutable). On peut faire recompiler Emologus en utilisant la commande suivante:

```
Process Findspace = Runtime.getRuntime().exec
                    (“compile_enfants”);
BufferedReader Resultset = new
     BufferedReader(new InputStreamReader
     (Findspace.getInputStream()));
```

Le mot « *compile_enfants* » est le nom du fichier qui contient les commandes pour compiler les applications Emologus, ci-dessous sont les commandes qui existent dans ce fichier :

Rekayasa ulang..., Agus Mulyanto, FT UI, 2012

```
"pmc cs_Emotirob.pm lex_Emotirob.pm translate1_
     Emotirob.pm jc_Emotirob.pm
     translate2_Emotirob.pm ce_Emotirob.pm
     contexte_Emotirob.pm main_Emotirob.pm -g -o
     emo_enfants"
```

`pmc` = Le command pour compiler l'application λProlog

`cs_Emotirob.pm(*.pm)` = Les modules dans Emologus

`-g –o` = Les options pour compiler dans λProlog

`emo_enfants` = Le nom du fichier résultant

A partir des huit modules ci-dessus, nous l'emballons en deux interfaces, à savoir l'interface pour afficher les résultats du traitement des données de sortie de l'Emologus (la lemmatisation, la segmentation et le calcul les valeurs d'émotions) et l'interface pour gérer les données des lexiques.

a. Interface du traitement des données de sortie de l'Emologus

Cette interface affiche les données des modules de traitement des données lemmatisation, module de segmentation, module de compréhension, module de calcul de la valeur émotionnelle, et module d'exécution / lancement des applications Emologus. Voici un affichage de la figure 4.6 l'impression d'écran d'interface du traitement des données de sortie de l'Emologus.

Rekayasa ulang..., Agus Mulyanto, FT UI, 2012

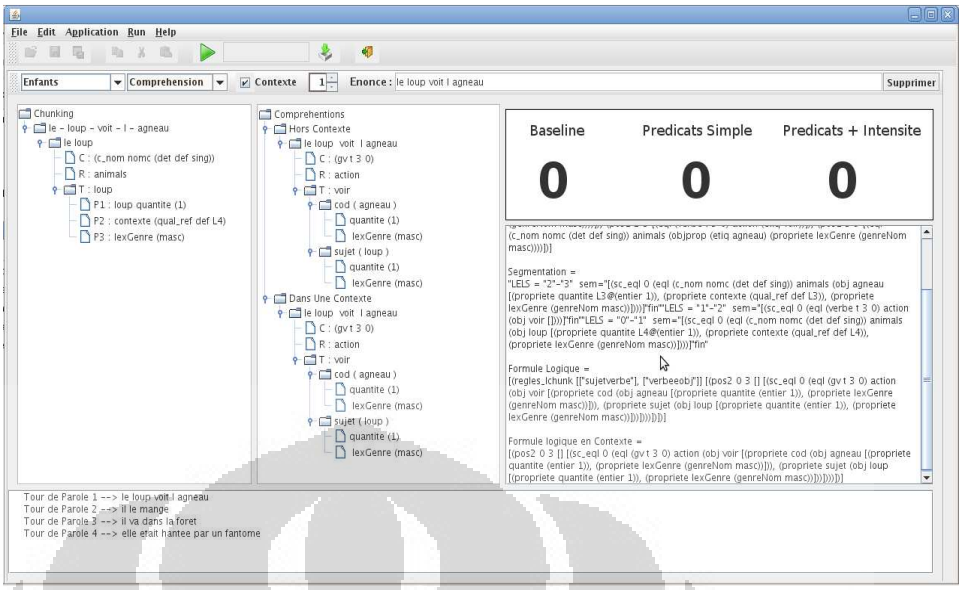


Figure 4.6. L'interface pour affiche les résultats d'Emologus

b. Interface pour gérer les données des lexiques

Cet Interface est utilisée pour gérer les lexiques de données avec sa définition, tels qu'ajouter, supprimer et modifier. La figure 13 sur monter l'impression d'écran de cette interface.

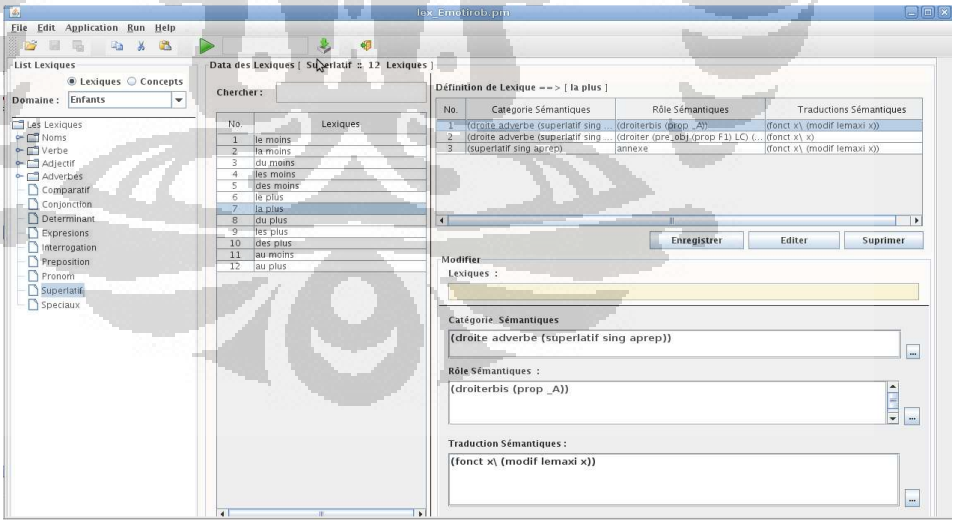


Figure 4.7 L'interface pour gérer les données des lexiques

4.4.2. Base de données lexicale

Les lexiques sont une base de données de EmoLogus où chaque lexique est défini par un ou plusieurs de représentation sémantique sont représentés par trois éléments qui sont appelle « triplets C; R; T ». Maintenant, les lexiques d'EmoLogus ont deux domaines d'ontologies « le domaine du renseignement touristique et la contes » qui sont regroupés dans un fichier « lex_emotirob.pm ». Ce lexique est trop gros et surtout, certaines définitions y entrent en concurrence.

Nous fournissons des solutions pour résoudre ce problème de la manière suivante : le lexique est séparé en deux domaines « le domaine du renseignement touristique » et « le domaine des contes » comme sur monter à la figure 4.8 Pour chaque domaine, on intègre le lexique général du français et le lexique de spécialité du domaine.

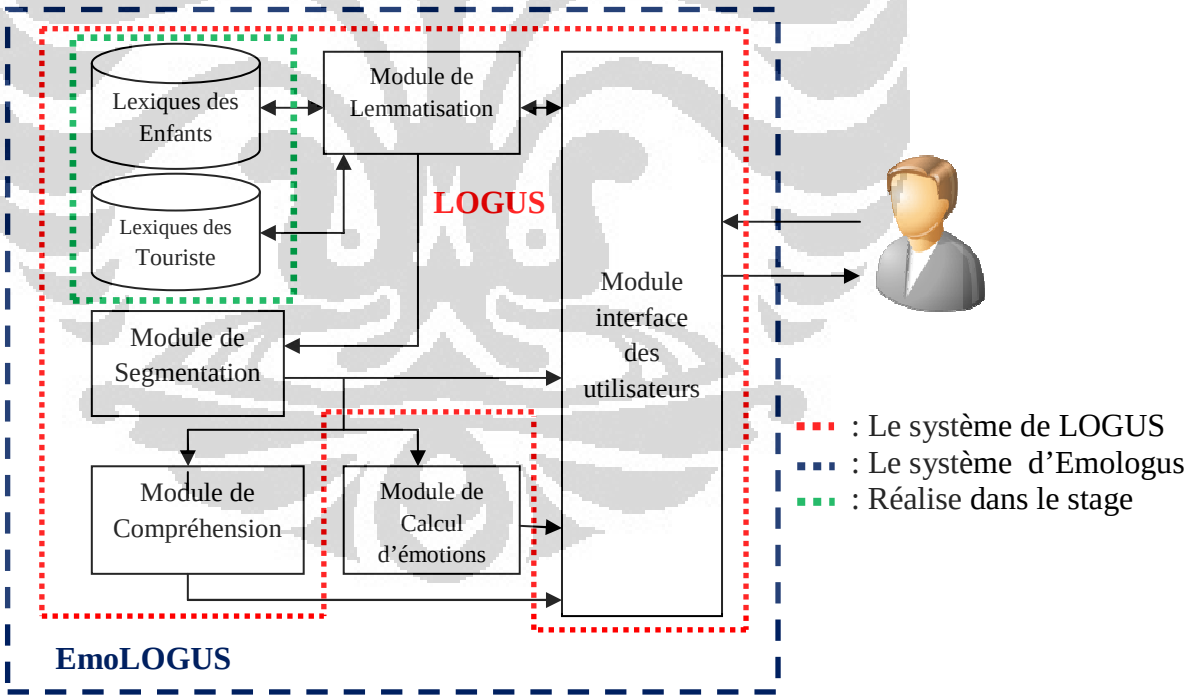


Figure 4.8 La structure de nouveau système d'Emologus

Rekayasa ulang..., Agus Mulyanto, FT UI, 2012

Ensuite, nous avons regroupé les entreés du lexique par catégorie syntaxique (le nom, l'adverbe, les verbes, etc.) et ensuite nous les avons triées par ordre alphabétique. Suivant les données de lexique d'Emologus dans deux domaines base sur ses catégorie:

1. *Noms* : Dans cette catégorie, sont de stockés es noms en conformité avec leur domaine d'ontologie respectif. Par exemple dans domaine les enfants on trouve les mots tels que *chat, rat, fantôme*, etc. et dans domaine renseignement touristique on trouve des mots tels que tels que *chambre, toilette*, etc.
2. *Verbe* : Cette catégorie de stockées les mots des verbes qui est divisé en 4 sous-catégories à savoir l'infinitif, le verbe du première personne, le verbe du deuxième personne, le verbe du troisième personne, et le groupe verbal. Groupe verbal correspond à une combinaison de deux mots qui sont habituellement constitués de sujet et le prédicat. Par exemple : « *je arrive, j'ai, il y a,* etc. »
3. *Adjectif* : Cette catégorie de stockées les mots tels que « *supplémentaire, suivante, un seul,* etc.»
4. *Superlatif* : Cette catégorie de stockées les mots tels que « *le moins, le plus, etc.* »
5. *Comparatif* : Cette catégorie de stockées les mots tels que « *inférieur, supérieur, etc. »*
6. *Conjonction* : Cette catégorie de stockées les mots tels que *« si, à condition que, si peut être, etc.* »
7. *Déterminant* : Cette catégorie de stockées les mots tels que *«son, leur, ses, cette, du, etc. »*
8. *Interrogation* : Cette catégorie de stockées les mots tels que *«quel, laquelle, comment, etc. »*
9. *Préposition* : Cette catégorie de stockées les mots tels que « *pas sans, avec, jusque vers, etc.* »
10. *Pronom* : Cette catégorie de stockées les mots tels que « *celui, je, tu, elle, il, nous, vous, ils, elles, etc. »*

Rekayasa ulang..., Agus Mulyanto, FT UI, 2012

11. *Adverbes* : Cette catégorie de stockées les mots tels que *« aujourd'hui, avant hier, demain, etc. »*
12. *Expressions* : Cette catégorie de stockées les mots tels que « *cette heure, jusque là bas, autour, à proximité, etc.* »

Rekayasa ulang..., Agus Mulyanto, FT UI, 2012

Chapitre 4
CONCLUSION

Le processus de réingénierie de l'application Emologus nécessitait un autre langage de programmation pour créer une interface graphique en raison de notre connaissance que le langage de programmation λProlog n'a pas les outils pour faire une interface graphique qui s'exécute sur le système d'exploitation Linux. Pour cette raison, dans ce stage, nous utilisons le langage de programmation Java pour la conception l'interface graphique des Emologus. Cette interface est utilisée pour traiter les données de sortie et les données de lexiques l'Emologus en une forme l'interface graphique facile à comprendre et à analyser. En plus de la facilité de l'analyse des définitions des mots des lexiques, nous séparons les données des lexiques basées sur deux domaines d'ontologie à savoir le domaine des enfants et le domaine du renseignement touristique.

Enfin, après ce stage, j'ai une connaissance sur les processus de compréhension du langage et le processus de calculer la valeur des émotions qui y sont contenus. Basée sur l'expérience dans ce stage, je sais comment utiliser cette application pour d'autre l'application.

Rekayasa ulang..., Agus Mulyanto, FT UI, 2012

BIBLIOGRAPHIES

[1] Antoine J.-Y., Le Tallec M., et Villaneau J., " *Evaluation de la détection des émotions, des opinions ou des sentiments : dictature de la majorité ou respect de la diversité d'opinions ?*", TALN 2011, 2011.

[2] Antoine J.-Y., Jérôme G, et et Antoine J.-Y., "*Quand le TAL robust s'attaque au langage parlé : analyse incrémentale pour la compréhension de la parole spontane*", TALN 2003, 2003

[3] Le Tallec M., et Duhaut D., "*EmoLogus: presentation of a Linguistically based Model for Emotion Detection and adaption to another context*", IEEE 2011, 2011.

[4] Le Tallec M., Antoine J.-Y., Le Tallec M., et Villaneau J., Savary A., Arielle S.-V., "*Détection des émotions à partir du contenu linguistique d'énoncés oraux : application à un robot compagnon pour enfants fragilisés*", TALN 2011, 2011.

[5] Le Tallec M., Antoine J.-Y., Le Tallec M., et Villaneau J., Savary A., Arielle S.-V., "*Détection hors contexte des émotions à partir du contenu linguistique d'énoncés oraux : le système EmoLogus*", TALN 2010, 2010.

[6] Meurs M.-J., Duvert F., Béchet F., Lefévre F., et Mori R., "*Semantic Frame Annotation on the French MEDIA corpus*", Framework Research Programme of the European Union (EU).

[7] Sébastien S.-A., Jost C., Le-Pévédic B., et Duhaut D., "*Dynamic behaviour conception for EmI companion robot*", ISR / ROBOTIK 2010, 2010.

[8] Sébastien S.-A. Le-Pévédic B., et Duhaut D., "*Experimentation to evaluate EmotiRob interaction model*", IEEE 2009, 2009.

[9] Sébastien S.-A. Le-Pévédic B., et Duhaut D., *"EmotiRob: an emotional interaction model"*, IEEE 2008, 2008.

[10] Sébastien S.-A. Le-Pévédic B., et Duhaut D., *"Building emotions with 6 degrees of freedom"*, IEEE 2007, 2007.

[11] Shibata Takanori, Wada Kazuyoshi, et Tanie Kazuo, "*Tabulation and Analysis of Questionnaire Results of Subjective Evaluation of Seal Robot in Japan, U.K., Sweden and Italy"*, IEEE 2004, 2004.

[12] Shibata Takanori, Inoue Kazuyoshi, et Irie Robert, "*Emotional Robot for Intelligent System – Artificial*", IEEE 1996, 1996.

Rekayasa ulang..., Agus Mulyanto, FT UI, 2012

[13] Villaneau J., Ridoux O., et Antoine J.-Y., *"LOGUS : compréhension de l'oral spontané"*, RSTI - RIA, 2004.

[14] Villaneau J., *"Contribution au traitement syntaxico-pragmatique de la langue naturelle parlée : approche logique pour la compréhension de la parole"*, thése à UBS, 2003.

[15] Wada Kazuyoshi, *"Development and Preliminary Evaluation of a Caregiver's Manual for Robot Therapy using the Therapeutic Seal Robot Paro"*, IEEE 2010, 2010

[16] Wada Kazuyoshi, Shibata Takanori, Musha Toshimitsu,et Kimura Shin, *" Robot Therapy for Elders Affected by Dementia"*, IEEE 2008, 2008

[17] Wada K. et Shibata T., *"Robot Therapy in a Care House - Change of Relationship among the Residents and Seal Robot during a 2-month Long Study"*, IEEE 2007, 2007.

[18] Wada K., Shibata T., Saito T., et Tanie K. , *"Psychological and Social Effects in Long-Term Experiment of Robot Assisted Activity to Elderly People at a Health Service Facility for the Aged"*, IEEE 2004, 2004.

.

Rekayasa ulang..., Agus Mulyanto, FT UI, 2012

ANNEXES

La Programmation Logique λProlog

Cette annexe veut présenter brièvement et aussi simplement que possible le langage λProlog qu'est utilise pour implémenter de l'EmoLOGUS. Les majorités des matériaux à l'annexe A cité de la thèse de Jean Villenau, 2002.

A. Introductions

Prolog et λProlog sont deux langages de la famille des langages relationnels que l'on appel aussi langages de programmation logiques. Le λProlog est une extension double de Prolog, au niveau des termes et au niveau des formules (Belleannée et al. 1999). Les termes de lambda-Prolog sont les lambda-termes simplement typés du lambda-calcul de Church. Les formules de lambda-Prolog sont les formules héréditaires de Harrop, une extension des clauses de Horn. Grâce à ces deux extensions complémentaires, λProlog permet de manipuler l'abstraction au niveau symbolique. En effet, les abstractions peuvent être traversées par les parcours inductifs sur la structure des termes. Le langage est fortement typé.

D'une part deux nouveaux connecteurs, l'implication et le quantificateur universel, permettent d'écrire des buts non permis en Prolog. D'autre part, les termes du premier ordre de Prolog sont remplacés par des lambda-termes typés. Les différences entre le prolog et lambda prolog à savoir [Yves Bekkers et al 2011] :

	Prolog	**λ-Prolog**
Formule Logique	Clauses de Horn	Formules Héréditaires de Harrops
Termes	Termes du premier ordre	Termes simplement typés du λcalcul
Unification	Termes du premier ordre	Résolution de contraintes sur les termes simplement typés du λcalcul

Rekayasa ulang..., Agus Mulyanto, FT UI, 2012

B. Les formules de λProlog

Les *formules* qui permettent de représenter des énoncés du monde réel.

Lambda-Prolog utilise les formules héréditaires de Harrop pour représenter les énoncés. Les formules héréditaires de Harrop sont des formules de Harrop héréditaires où les termes sont des λ-termes simplement typés. Elles constituent la logique de référence du langage de programmation. Ce fragment logique présente un caractère inhabituel du fait de l'indécidabilité de son unification. L'ensemble des *formules héréditaires de Harrop* (désignées par D) et celui des buts (désignés par B) sont définis par les grammaires suivantes, mutuellement récursives, où A désigne une formule atomique :

$$D ::= A \mid D \wedge D \mid B \Rightarrow A \mid \forall \chi D$$

$$B ::= A \mid B \wedge B \mid D \Rightarrow B \mid \forall \chi B \mid \exists \chi B$$

Où : $\vee$ = le connecteur « ou »

$\exists$ = le quantificateur «il existe»

$\wedge$ = le connecteur «et»

$\Rightarrow$ = le connecteur «implique»

$\forall$ = le quantificateur «quel que soit» sans restriction

Les formules ainsi définies satisfont à la définition d'un langage de programmation logique au sens donné par Miller, Nadathur et Scedrov : λ-Prolog. Par rapport à Prolog, la sémantique opérationnelle de λProlog introduit dans le corps des clauses l'implication, la quantification universelle, la quantification existentielle et la disjonction.

Disjonction à droite (V^i_B) : elle n'induit pas de nouvelles possibilités par rapport à Prolog ou la disjonction à droite se traduit simplement par la conjonction de deux clauses $((P_1 \vee P_2)) \Rightarrow Q) \equiv (P_1 \Rightarrow Q) \wedge (P_2) Q))$.

Rekayasa ulang..., Agus Mulyanto, FT UI, 2012

Quantification existentielle à droite (∃B) : dans Prolog, les variables sont quantifiées universellement au niveau des clauses et donc existentiellement au niveau des buts (si Q ne contient pas d'occurrence libre de χ, $\forall\chi$(P(χ) ⇒ Q) ≡ ($\exists\chi$ P(χ) ⇒ Q)). Donc, sur ce point encore, il n'y a pas de nouveauté par rapport à Prolog.

Implication à droite (⇒B) : il s'agit là de l'implication intuitionniste : on n'a pas (D ⇒ B ≡ ¬D∨B) et on ne peut pas prouver ¬D pour prouver D⇒B. D'après la règle des séquents, obtenir la preuve de (D ⇒B) consiste à ajouter l'hypothèse D au programme pour obtenir la preuve du but B. Cette implication permet par exemple d'ajouter dynamiquement des hypothèses, sans utiliser, comme en Prolog, des prédicats extra-logiques (*assert* ou *retract*).

Quantification universelle à droite (∀B) : pour démontrer ($\forall\chi$ B(χ)), on démontre B(c) où
c'est une nouvelle constante qui ne doit apparaître ni dans B, ni dans P. Il s'agit donc d'une démonstration de B(χ) qui doit être la même pour tous les χ concernés et non de la démonstration de B(x) pour chacune des valeurs possibles de la variable χ (démonstration *intentionnelle*). Cette sémantique opérationnelle est complètement inconnue dans le langage Prolog.

C. Les termes de λProlog

La programmation logique repose sur la définition de prédicats ou relations portant sur des objets syntaxiques appelés des termes. Les termes sont des représentations symboliques des objets de ce monde réel. Les formules permettent de représenter les relations. Motivés par le besoin d'introduire des variables de prédicats pour résoudre des problèmes de démonstration automatique, Miller et Nadathur proposent également une extension des termes manipulés par Prolog (Miller and Nadathur, 1986). Cette idée simple ignore que le typage est un élément essentiel de λProlog. Il joue le rôle de « *spécification partielle des programmes* » (Ridoux, 1998) et permet de détecter un grand nombre d'erreurs à

Rekayasa ulang..., Agus Mulyanto, FT UI, 2012

la compilation. Par contre, il rend impossible certaines pratiques courantes (et discutables) en Prolog telles que par exemple l'utilisation de listes hétérogènes sans autres précautions.

Les termes du λ-calcul permettent la représentation des fonctions par les règles qui les représentent (Lalement, 1990). Si $\mathcal{C}_t$ et $\mathcal{V}_t$ désignent respectivement les identificateurs de constantes et de variables de type t, les règles de formation des λ-termes simplement typés correspondent aux trois règles de grammaire suivantes (Jeanne VILLANEAU, 2002):

1. $\wedge_t ::= \mathcal{C}_t / \mathcal{V}_t$ les constantes et les variables sont des λ-termes de type t.
2. $\wedge_{t1 \to t2} ::= \lambda \mathcal{V}_t . \wedge_{t2}$ si χ est une variable de type et un λ-terme de type t_2, alors $\lambda\chi$.L est un λ -terme de type($t1 \to t2$)(appelé *abstraction*).
3. $\wedge_{t2} ::= (\wedge_{t1 \to t2} . \wedge_{t1})$: si *L1* et *L2* sont des λ-termes de types respectifs *t1* $\to$ *t2* et *t1*, alors (*L1* et *L2*)est un λ-terme de type *t2*. Il est appelé *application* de *L1* à *L2*.

Les λ-termes sont classiquement munis des trois équivalences suivantes :

- α- équivalence : elle correspond au renommage des variables.

  $$\lambda\chi.E \equiv_\alpha \lambda y.\mathrm{E}\,[\chi \leftarrow y\,]$$

  La condition pour que ce renommage soit licite est que y n'appartienne ni aux variables libres ni aux variables liées de *E*.

- β - équivalence : elle correspond à l'application d'une fonction à l'un de ses arguments :

  $$(\lambda\chi\ (E)F\ ) \equiv_\beta \mathrm{E}\,[\chi \leftarrow \mathcal{F}\,]$$

- η - équivalence : une fonction est déterminée par l'image qu'elle donne d'un argument quelconque (extensionalité fonctionnelle)

  $$\Lambda\chi(E\ \chi) \equiv_\eta \mathrm{E}$$

  à condition que χ ne soit pas une variable libre de E.

Rekayasa ulang..., Agus Mulyanto, FT UI, 2012

On appelle α-*conversion* et *(β - η)-réductions* les règles de réécriture qui résultent de l'orientation de la gauche vers la droite de ces équivalences. On appelle *(β - η)-rédex* les instances des parties gauches de ces équivalences. On appelle terme *(β - η)-normal* un terme qui ne contient pas de *(β - η)-rédex*. La β (-η)-équivalence de deux termes simplement typés peut se vérifier par la β (-η)-convertibilité vers un même terme β (-η)-normal. C'est un problème décidable.

Un λ-terme peut contenir plusieurs β (-η)-rédex. Dans ce cas, plusieurs règles de conversion sont candidates à s'appliquer. La théorie du _-calcul non-typé montre que l'ordre de traitement des β (-η)-rédex est indifférent pour la forme normale à trouver, mais qu'il ne l'est pas pour la terminaison du calcul. Au contraire, le λ-calcul simplement typé a la propriété de la normalisation forte : on peut atteindre une forme normale en un nombre fini d'étapes quel que soit le choix du rédex à réduire à chaque étape.

Base sur de la grammaire de Montague : à chaque règle syntaxique correspond une règle sémantique (Montague, 1974). Elle suppose que l'on peut appliquer le principe de compositionnalité du sens, suivant lequel on peut calculer le sens d'une expression (syntaxiquement bien formée) en composant le sens des expressions qui la constituent. La représentation sémantique des énoncés est une formule de la logique des prédicats du premier ordre. Par exemple, la formule logique qui correspond à l'énoncé «*Pierre dort*» est (*dort Pierre*) ou « *dort*» est un prédicat d'arité 1 et « *Pierre*» une constante. Les déterminants correspondent à des quantifications comme dans l'énoncé «*Pierre mange une pomme*» traduit par $\exists x$ ((*pomme* x) ∧ (*manger Pierre* x)) où « *pomme*» est un prédicat d'arité 1 dont l'interprétation est (*pomme* x) si et seulement si x est une rat et où mange un prédicat d'arité 2 d'interprétation (*mange* x y) si et seulement si (x *mange* y).

Dans les grammaires AB, on se donne un ensemble fini P de catégories de base, par exemple {sn; nom_c; S} (sn pour syntagme nominal et nom_c pour nom commun) où S joue un rôle particulier, analogue à celui joué par le symbole non-terminal initial des grammaires génératives. On définit inductement l'ensemble des catégories par :

Rekayasa ulang..., Agus Mulyanto, FT UI, 2012

1. Les catégories de base sont des catégories.
2. Si C_1 et C_2 sont des catégories, alors $C_1 \backslash C_2$ et C_2/C_1 sont des catégories.

Un lexique associe à chaque mot de la langue une ou plusieurs catégories. Les deux règles d'association entre les catégories correspondent à des règles de calcul sur les fractions (produit non commutatif) $x\ (x\backslash y) \rightarrow y$ et $(y/x)\ x \rightarrow y$. La règle $x\ (x\backslash y)$ signifie qu'un mot de catégorie $(x\backslash y)$ peut prendre à sa gauche un mot de catégorie x pour former un composant de catégorie y et la règle 2 qu'un mot de catégorie (y/x) peut prendre à sa droite un mot de catégorie x pour former un composant de catégorie y.

Le langage engendré par cette grammaire est l'ensemble des suites de mots $m_1 m_2 \ldots m_n$ tel que, pour chaque mot m_i, il existe une catégorie c_i telle que $c_1 c_2 \ldots c_n \rightarrow^* S$. Ainsi par exemple, l'énoncé *« Pierre mange une pomme »* est reconnu grammatical par :

« Pierre »	**« mange »**	**« une »**	**« pomme »**	
sn	*sn\(S/sn)*	*sn/nom_c*	*nom_c*	→
sn	*sn\(S/sn)*		*sn*	→
	S/sn		*sn*	→
		S		

Alors, si on veut représenter ces phrases sur les catégories syntaxiques, les types et les λ-termes associés on peut utiliser une méthode de calculassions de Lambek et sémantiques de Montague. Le calcul de Lambek définit les deux types élémentaires ***e***, le type des entités et ***t***, le type des énoncés (valeur de vérité). Ainsi par exemple, $((e \rightarrow t) \rightarrow (e \rightarrow (e \rightarrow t)))$ est un type. Pour simplifier l'écriture, le parenthésage est sous-entendu à droite, et le type précédent peut s'écrire $((e \rightarrow t) \rightarrow e \rightarrow e \rightarrow t)$. Un terme prédicatif d'arité 4 a donc par exemple pour type $(e \rightarrow e \rightarrow e \rightarrow e \rightarrow t)$ et un terme fonctionnel d'arité 3 pour type $(e \rightarrow e \rightarrow e \rightarrow e)$.

À chaque définition d'un mot m du langage est associé un type et un λ-terme. Par exemple, le nom propre *« Pierre »* est un syntagme nominal. Le λ-terme qui lui est associé correspond à l'ensemble des prédicats d'arité 1 qui

Rekayasa ulang..., Agus Mulyanto, FT UI, 2012

peuvent être appliqués à l'entité (de type e) Pierre pour former la représentation sémantique de l'énoncé : *λP.(P Pierre)*. Comme *P* est un prédicat de type *(e → t)*, il s'agit d'une abstraction de type *((e → t) → t)*.

La grammaticalité d'un énoncé est démontrée par un calcul de la logique ntuitionniste, grâce au calcul de Lambek. Grâce à l'isomorphisme de Curry-Howard, ce calcul des séquents correspond à un λ-terme simplement typé. La β-réduction de ce λ-terme permet d'obtenir la représentation sémantique de l'énoncé. Ci-dessous sont les exemples utilisation la méthode de calcul de Lambek et sémantiques de Montague d'obtenir les λ-termes de l'énonces.

La phrase 1 : **«Pierre dort»**

Où la catégorie de *Pierre* = *sn* et *dort*=*sn**s*

La grammaticalité de cet énoncé est assurée directement par le séquent (règle e/) :

$$\frac{\text{Pierre} \vdash sn \quad \text{dort} \vdash sn \backslash s}{\text{Pierre}, \text{dort} \vdash s}\, e\backslash$$

Appliquée à un calcul sur les types, cette démonstration devient :

$$\frac{\text{Pierre} \vdash (e \to t) \to t \quad \text{dort} \vdash e \to t}{\text{Pierre}, \text{dort} \vdash t}$$

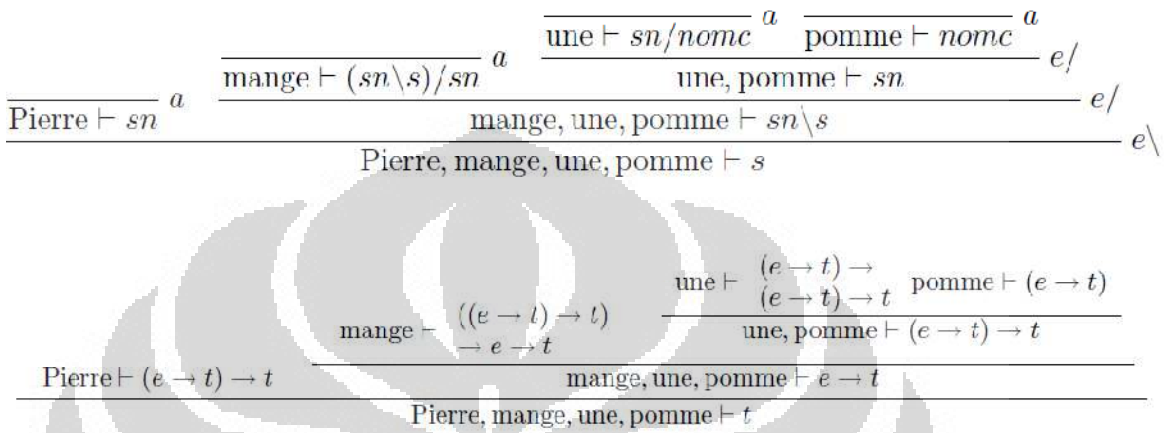
Le λ-terme qui correspond à la représentation sémantique de l'énoncé s'obtient par λ-réduction de la composition des λ-termes qui correspondent à cette démonstration :

$$(\wedge_P \wedge_D) \equiv_\beta (\lambda P.(P\ \text{Pierre})\ \lambda x.(\text{dort}\ x))$$
$$\equiv_\beta (\lambda P.(P\ \text{Pierre})\ \text{dort})$$
$$\equiv_\beta (\text{dort}\ \text{Pierre})$$

Rekayasa ulang..., Agus Mulyanto, FT UI, 2012

La phrase 2 : **« Pierre mange une pomme »**

La preuve suivante (en déduction naturelle) démontre la grammaticalité de l'énoncé.



Le λ-terme associé à cette démonstration s'obtient par la composition qui correspond à la
peuve : $(\wedge_P (\wedge_M (\wedge_U \wedge_{Po})))$ par β-réduction :

$(\wedge_P (\wedge_M (\lambda P\ \lambda Q{:}(\exists p\ (P\ p) \wedge (Q\ p))\ \lambda x{:}(pomme\ x))))$

$\equiv_\beta (\wedge_P (\wedge_M\ Q{:}(\exists p\ (pomme\ p) \wedge (Q\ p))))$

$\equiv_\beta (\wedge_P (\lambda C\ \lambda x{:}(C\ (mange\ x))\ \lambda Q{:}(\exists p\ (pomme\ p) \wedge (Q\ p))))$

$\equiv_\beta (\wedge_P (\lambda x{:}(\lambda Q{:}(\exists p\ (pomme\ p) \wedge (Q\ p))\ (mange\ x))$

$\equiv_\beta (\wedge_P\ \lambda x{:}(\exists p\ (pomme\ p) \wedge (mange\ x\ p)))$

$\equiv_\beta (\lambda x.(x\ Pierre)\ \lambda x{:}(\exists p\ (pomme\ p) \wedge (mange\ x\ p)))$

$\equiv_\beta \exists p\ (pomme\ p) \wedge (mange\ Pierre\ p)$

Rekayasa ulang..., Agus Mulyanto, FT UI, 2012

mot	cat. synt.	type	λ-terme	Id.
« *Pierre* »	sn	$(e \to t) \to t$	$\lambda P.(P\ Pierre)$	Λ_P
« *dort* »	sn\s	$e \to t$	$\lambda x.\ (dort\ x)$	Λ_D
« *mange* »	(sn\s)/sn	$((e \to t) \to t) \to e \to t$	$\lambda C \lambda x.(C\ (mange\ x))$	Λ_M
« *une* »	sn/nc	$(e \to t) \to (e \to t) \to t$	$\lambda P.\lambda Q.(\exists x\ (P\ x) \wedge (Q\ x))$	Λ_U
« *pomme* »	nc	$(e \to t)$	$\lambda x.(pomme\ x)$	Λ_{Po}

D. Un exemple de la programmation λProlog

Les λHHG sont le résultat de la transposition à λProlog du principe des DCG (Le Huitouze et al.1993; Coupet-Grimal and Ridoux, 1995) ; elles intègrent les formules de Harrop au niveau grammatical et pour le calcul des attributs. Par ailleurs, les λ-termes permettent une construction directe de la structure sémantique qui correspond à l'analyse de l'énoncé. Par exemple, on se donne la (minuscule) grammaire suivante :

< phrase > *::= < sn >< gv >*
< sn > *::= < nomp > | < det >< nomc >*
< gv > *::= < verbe_trans >< sn >*
< gv > *::= < verbe_intrans >*
< nomp > *::= "Pierre"*
< nomc > *::= "pomme"*
< det > *::= "une"*
< verbe_trans > *::= "mange"*
< verbe_intrans > *::= "dort"*

On veut que la représentation sémantique en _Prolog de l'énoncé *« Pierre dort »* soit (dormir pierre) et que celle associée à *« Pierre mange une pomme »* soit (existe Xn(et (pomme X) (manger pierre X))). On donne d'abord la déclaration des termes qui interviennent dans ce langage cible :

kind e type. % le type des entités
type pierre e.
type (dormir, pomme) e → o. % prédicats d'arité 1

Rekayasa ulang..., Agus Mulyanto, FT UI, 2012

type manger e → e → o. % prédicat d'arité 2

type et o → o → o. % concaténation de valeurs booléennes

type existe (e → o) → o. % quantification d'un prédicat d'arité 1

Les non-terminaux donnent lieu aux déclarations suivantes : les mots sont représentés par des listes en différence de chaînes de caractères (*(list string) -> (list string)*), la première partie de la déclaration donne le type du _-terme associé au non-terminal.

#define stringt ((list string) → (list string) → o)

type phrase o → stringt.

type (sn, nomp) ((e → o) → o) → stringt.

type nomc (e → o) → stringt.

type (gv, verbe_intrans) (e → o) → stringt.

type verbe_trans (e → e → o) → stringt.

type det ((e → o) → (e → o) → o) → stringt.

Chacune des règles de grammaire contient dans son attribut la composition des fonctions sémantiques correspondantes : (& est le connecteur de concaténation, --> est le signe de la production).

phrase (SN V) --> sn SN & gv V.

sn N --> nomp N.

sn (A NC) --> det A & nomc NC.

gv V --> verbe_intrans V.

gv x\ (N (V x)) --> verbe_trans V & sn N.

Les règles des terminaux donnent en attributs la fonction sémantique associée à chaque mot.

verbe_intrans dormir --> $["dort"].

verbe_trans manger --> $["mange"].

nom_propre x \ (x pierre) --> $["Pierre"].

Rekayasa ulang..., Agus Mulyanto, FT UI, 2012

nom_commun pomme --> $[“pomme”].
det x\ (y\ (existe p\ (et (x p) (y p)))) --> $[“une”].

Ce programme permet d’obtenir directement les traductions sémantiques attendues, la β-réduction étant directement assurée par le langage de programmation. Par exemple, l’énoncé *« Pierre dort »* rend *(dormir pierre)* obtenu par β-réduction de *(x\(x pierre) dormir)*. Le programme peut fonctionner également comme générateur de phrases. L’exemple précédent est très élémentaire et utilise essentiellement les λ-termes et la β-réduction. S. Coupet-Grimal et O. Ridoux donnent des exemples d’application des λHHG au traitement automatique des langues qui utilisent les nouveaux connecteurs correspondant aux formules de Harrop (Coupet-Grimal and Ridoux, 1995).

Rekayasa ulang..., Agus Mulyanto, FT UI, 2012