La coédition langue UNL pour partager la révision entre langues d'un document multilingue

Wang-Ju Tsai

THESE

présentée et soutenue publiquement par

TSAI Wang-Ju

pour obtenir le titre de

DOCTEUR DE L'UNIVERSITÉ JOSEPH FOURIER – GRENOBLE 1

Spécialité

INFORMATIQUE

LA COEDITION LANGUE_UNL

POUR PARTAGER LA REVISION ENTRE LANGUES

D'UN DOCUMENT MULTILINGUE

---

9 juillet 2004

Jury :

Mme.	Marie-France	BRUANDET	Président
M.	Patrice	POGNAN	Rapporteur
M.	Paul	SABATIER	Rapporteur
M.	Marc	DYMETMAN	Examinateur
M.	Gilles	SÉRASSET	Examinateur
M.	Christian	BOITET	Directeur

THÈSE PRÉPARÉE AU SEIN DU GETA, LABORATOIRE CLIPS (IMAG, UJF, INPG & CNRS)

Résumé

Étant donnée la demande croissante en communication multilingue, il est de plus en plus nécessaire de créer et de maintenir des documents multilingues, pour les entreprises internationales comme pour les internautes. Pourtant, le problème principal reste : le coût de traduction et de révision d'un document multilingue croît linéairement en fonction du nombre de langues. Pour le résoudre, nous proposons de produire ces documents multilingues par traduction automatique (TA), de partager le travail de révision entre les langues, et de réviser incrémentalement, à la demande et en mode coopératif.

Notre solution est fondée sur l'utilisation d'un système de TA à « pivot », et reprend l'idée de « coédition » utilisée dans certains systèmes de génération multilingue. Pour des raisons développées en détail, UNL (Universal Networking Language) semble le meilleur langage pivot pour un tel système. Dans notre approche, l'utilisateur peut non seulement éditer directement le texte, mais aussi « coéditer » le graphe à travers le texte. Pour cela, une heuristique construit automatiquement une correspondance fine entre le texte et le graphe UNL en n'utilisant que des ressources disponibles gratuitement pour beaucoup de langues (segmenteurs, lemmatiseurs, dictionnaires). Pour chaque fragment de texte ainsi relié au graphe, on peut construire un menu dont chaque item est formé d'une annotation dans le texte et d'une action sur le graphe. Le graphe modifié peut être ensuite déconverti dans plusieurs langues, qui bénéficient toutes des corrections effectuées. Une maquette permet de démontrer un scénario dans lequel l'utilisateur alterne entre lecture (monolingue) et coédition.

Mots-Clés : Traduction Automatique, partage de révision, langage pivot, interlingua, coédition, UNL, correspondances entre structures, génération multilingue.

Abstract

As the demand for multilingual communication increases, the need to generate and to maintain multilingual documents becomes more and more important, for both international firms and ordinary Internet users. However, the main problem remains : the cost of translation and postediting of multilingual documents increases linearly with the number of the languages involved. To solve this problem, we propose to produce multilingual documents by machine translation (MT), to share the task of revision among languages, and to postedit incrementally on demand and in cooperative mode.

Our solution is based on using a "pivot" MT system, and building on the idea of the "co-edition" as used in some multilingual generation systems. As detailed in the thesis, UNL (Universal Networking Language) seems to be the best pivot language for such a system. Users can not only directly edit the text, but also "co-edit" the graph through the text. In order to achieve this, a heuristic method is proposed to construct automatically a fine-grained correspondence between the text and the UNL graph by using only freely available resources for many languages (segmenters, lemmatisers, and dictionaries). For each segment of the text linked to the graph in this way, we can construct a menu, in which each item consists of an annotation of the text and an action on the graph. The modified graph can then be deconverted into several languages, all of which benefit from the corrections. A prototype demonstrates a scenario where the user switches between reading mode (monolingual) and co-editiing mode.

Key words : Machine translation, postediting sharing, pivot language, interlingua, co-edition, UNL, correspondences between structures, multilingual generation.

Remerciements

En premier lieu, je remercie profondément le directeur de ma thèse, le professeur Christian BOITET, qui m'a toujours poussé jusqu'au bout et m'a toujours soutenu aux moments les plus difficiles. C'est lui qui m'a montré et appris la persistance et la précision indispensables pour être un chercheur. Je suis toujours impressionné par son exigence et sa passion pour la TA.

Je remercie mes rapporteurs, le professeur Paul SABATIER et le professeur Patrice POGNAN, qui ont accepté d'être rapporteurs de ma thèse à une période très chargée. Je remercie le professeur Marie-France BRUANDET et le professeur Marc DYMETMAN pour accepter d'être le président et l'examinateur de ma thèse.

Je remercie le professeur Etienne BLANC, qui m'a guidé dans la TA sur ARIANE et UNL. Je remercie aussi le professeur Gilles SÉRASSET, Mr. Youcef BEY, et Mr. Stéphane HELME pour leur aide et leur contribution à la programmation de la maquette.

Je remercie monsieur Hiroshi UCHIDA pour avoir inventé l'UNL, et toute la communauté UNL, surtout le professeur Igor BOGUSLAVSKY, le professeur Jésus CARDEÑOSA, et le professeur Irina PRODANOF, pour m'avoir aidé sur la déconversion du russe, de l'espagnol et de l'italien.

Je remercie aussi l'ensemble de l'équipe GETA qui m'a accueilli et aidé durant ces années à Grenoble. Merci à Mutsuko et à Aree pour m'avoir aidé à corriger le texte japonais et thaï. Et surtout merci à Karën, Christophe, Mathieu pour leur amitié.

Je remercie le professeur François TCHEOU, qui m'a accueilli chaleureusement quand je venais d'arriver à Grenoble, et m'a soutenu tout au long de mon séjour en France, et m'a toujours fait confiance.

Je tiens à remercier Mr. John Kent de Londres et Madame Christina Cross de Lodi, Californie, pour leur soutien psychologique, qui m'a beaucoup aidé à mieux me comprendre.

Enfin et surtout, mes remerciements vont à vous, ma famille à Taiwan, ma Grande-mère, mes parents et Yi-Chia, sans vos soutiens cette thèse n'aurait pas été possible. La conversation téléphonique hebdomadaire avec vous m'a été très importante et chère. Merci encore pour votre patience et votre écoute. Vous êtes toujours dans mon cœur.

I would like to thank Mr. John Kent from London and Ms. Christina Cross from Lodi, California, without your insights, encouragement, and long-term support, I wouldn't be able to come this far, and would probably still be entangled in the push-and-pull of my emotions. It is the dialogue with you that keeps me conscious and opens me up to the spiritual and psychological world. I appreciate a lot the tools and the lessons you brought me and hope that I can still keep on making the conscious choices in both scientific and psychological fields, stop jumping on one foot and find the keys which are out there in the dark, beyond the light of the lamp.

Table des matières

Liste des figures

Liste des tableaux

Introduction

Situation et motivations

Il est de plus en plus nécessaire de créer et de maintenir des documents multilingues. Nous pensons surtout aux entreprises internationales comme HP, Cisco, Bull, Aix, etc. qui ont le besoin de communiquer avec le grand public en plusieurs langues. Par exemple, HP a 200.000 notices en anglais sur son site web, et Cisco produit 40.000 pages de documents chaque mois en langues CJK (chinois, japonais, coréen). Pour le maintien de ces documents multilingues et la gestion de versions, A. Assimi [Assimi 00] a montré comment « réaligner » des documents parallèles décentralisés et leur appliquer ensuite sa méthodologie de production d'un nouvel original en langue source, et de traduction vers les langues cibles des parties modifiées.

Le problème général reste : aussi bien les traductions que les révisions ont un coût croissant linéairement en fonction du nombre de langues. Cela reste vrai même si on se limite, dans le cas de l'évolution de documents multilingues, à ne retraduire (et donc réviser) que les parties qui ont changé.

Ce que nous aimerions, c'est produire ces documents multilingues par la TA, et faire en sorte que le travail de révision puisse être partagé entre les langues, quels que soient le domaine et le contexte.

Nos trois idées principales sont :

> (1) Mutualisation et collaboration : les utilisateurs révisent sur Internet une bonne partie des textes de documents multilingues traduits par la machine. Nous visons la révision et l'amélioration de la communication multilingue écrite sur Internet, dans un domaine ouvert, où la qualité de traduction peut être non-professionnelle.
>
> (2) Révision à la demande : on ne révise pas tout, mais seulement le plus important, c'est-à-dire les endroits où l'utilisateur pense que cela en vaut la peine.
>
> (3) Partage de la révision entre les différentes langues : c'est le problème le plus difficile mais avec une grande économie potentielle.

Bien sûr, on ne peut pas garantir la qualité de la révision faite par un utilisateur quelconque, mais on peut limiter le type de correction si on construit un environnement « guidé ». Dans la pratique, il n'est pas nécessaire que la qualité d'un document traduit soit uniforme. C'est pourquoi nous proposons de faire la révision « à la demande ».

Il est clairement impossible de refléter les changements sur un fichier en langue L0 dans les fichiers en langues L1,… Ln automatiquement et fidèlement, sans une structure intermédiaire pour faire le pont, car il faudrait au moins un aligneur parfait à granularité très fine dans le cas simple d'un changement d'article ou de nom (et encore, en supposant que le genre et le nombre restent les mêmes dans chaque version Li). Dans le cas du remplacement d'un verbe par un autre verbe ayant un régime

différent dans une langue cible Li, il faudrait réanalyser la phrase en Li, la transformer en conséquence, et la regénérer sans introduire de nouvelle erreur ou imprécision, tout en gardant les améliorations manuelles éventuellement apportées lors de révisions précédentes. Ou bien, il faudrait disposer d'un système de TA plus que parfait, à savoir capable d'analyser l'énoncé modifié en L0, de le transférer, et de générer un énoncé aussi proche que possible de l'énoncé précédent en Li, toujours en supposant que celui-ci pourrait avoir été amélioré manuellement lors d'une étape précédente.

L'approche la meilleure et la plus simple nous semble être d'utiliser un interlingua formel IL et :

de répercuter les modifications de L0 vers l'IL,

de regénérer vers L1,… Ln depuis l'IL.

Il faudra cependant permettre des améliorations manuelles, car la forme interlingue ne sera pas toujours présente, ou pas assez améliorable par défaut d'expressivité, et les générateurs ne seront jamais parfaits.

Intérêt de notre travail

L'intérêt de notre travail est que cette nouvelle idée de correction d'une structure intermédiaire à travers une version textuelle pourra permettre d'améliorer les autres versions dans d'autres langues, et donc, pour la première fois dans l'histoire de la traduction, de « partager le travail de révision ».

Un autre point intéressant est que nous nous plaçons dans un cadre collaboratif, sur Internet. Ainsi, ce sont les lecteurs des documents qui détermineront les passages où la qualité est la plus importante, et les réviseront. D'où une troisième idée, celle d'une amélioration incrémentale et à la demande.

Enfin, nous utilisons la génération de texte (la plupart de temps limitée à des domaines restreints) dans le domaine général, et visons des utilisateurs ordinaires et pas seulement des experts.

La mise en œuvre de ces idées impose d'approfondir un certain nombre de points :

quelle « structure intermédiaire » choisir ? Après un étude assez large, notre choix s'est porté sur UNL (Universal Networking Language), langage d'hypergraphes linguistico-sémantiques décrivant des structures abstraites d'énoncés reflétés en anglais.

Comment faire modifier une structure intermédiaire de ce genre par des utilisateurs « naïfs » ? Nous proposerons une « coédition » de cette structure à travers un texte, c'est-à-dire une annotation d'éléments d'un texte provoquant les modifications désirées sur la structure, qui peut rester cachée, sauf dans un mode « expert ».

Pour réaliser une telle coédition à partir d'un couple (texte, structure), *comment établir une correspondance fine entre le texte et la structure,* sans disposer d'analyseur ni de générateur, ni a fortiori d'une spécification formelle de cette correspondance ? Nous introduirons là aussi une méthode originale fondée sur l'utilisation de ressources disponibles gratuitement pour beaucoup de langues.

Organisation de la thèse

Nous divisons cette thèse en quatre parties :

Partie A (Contexte et motivations) : nous commencerons par une étude de plusieurs systèmes de TA et de génération automatique de langue naturelle pour clarifier l'idée de « coédition ». Nous définirons nos critères, notre terminologie, et les aspects linguistiques et informatiques souhaitables dans un système de coédition. Nous décrirons aussi comment l'idée de coédition pourra s'intégrer dans un système d'information.

Partie B (Quel langage pivot choisir ?) : nous étudierons plusieurs systèmes existants qui ont exploité un interlingua, et concluons que l'interlingua qui nous convient le mieux est UNL. Nous donnerons nos raisons et encore plus de détails sur l'état courant de ce langage et du projet international de recherche organisé autour de ce langage. Nous décrirons un scénario d'un système de coédition utilisant UNL et comment construire un tel système, étant données les caractéristiques d'UNL.

Partie C (Étude des correspondances UNL-texte) : nous étudions divers modèles permettant de décrire la correspondance entre deux structures, et présentons notre algorithme heuristique pour créer la correspondance entre un texte et un graphe UNL

Partie D (Implémentation de la plate-forme SWIIVRE) : nous présentons la plate-forme que nous avons construite pour les expériences sur UNL et sur la coédition. Nous montrons aussi le résultat d'une maquette que nous avons réalisée.

A. Contexte et motivations

Introduction

Nous commençons cette partie en précisant le contexte dans lequel nous nous plaçons, - en bref, la communication multilingue écrite sur Internet - et les trois axes qui devraient permettre d'augmenter la qualité « utile » de cette communication, tout en en réduisant fortement les coûts : technique de partage de l'effort de révision par « coédition », mutualisation et bénévolat dans ce travail de révision, et diminution de l'effort à tous les stades par « amélioration à la demande ».

Nous cherchons ensuite à préciser quel type de « coédition » sera le plus adapté dans ce contexte. Pour cela, nous étudions un certain nombre de systèmes récents permettant de « coéditer » deux textes parallèles, ou plusieurs textes générés dans des langues différentes à partir d'une même structure interne, etc. Nous aurons ainsi une taxonomie des systèmes de coédition, menant à la définition d'une terminologie précise, ainsi qu'à une comparaison entre les différents types de systèmes.

Enfin, nous déterminons le type de coédition à employer pour la communication multilingue écrite sur Internet, et plus généralement les caractéristiques souhaitables pour un systèmes d'information multilingue organisé autour de ces idées.

1. Position du problème et motivation du paradigme de la coédition de textes multilingues

1.1 Problème de la TA «classique»

Puisque nous nous plaçons dans le contexte de la communication multilingue écrite sur Internet, il nous faut d'abord préciser de quel type de communication il s'agit, et du rôle que peut jouer la TAO sous ses différentes formes.

D'abord, nous visons aussi bien la communication professionnelle que privée. Dans le premier cas, il s'agit de rendre disponible à faible coût et à qualité « suffisante » aussi bien de la littérature « grise » comme des notices d'installation ou des aides en ligne que des manuels d'utilisation ou des pages web de musées et autres sites culturels. Dans le second, il peut s'agir de courriels, ou de petits documents, mais pas (pour l'instant) de dialogues ni de « tchats » pour lesquels il ne semble pas utile d'augmenter la qualité de traduction après coup (sauf peut-être pour établir des PV de discussions). En tout cas, nous supposons que, quelle que soit la méthode de traduction utilisée, le résultat n'est ni parfait ni totalement désambiguïsé.

Que peut-on attendre de la TAO « classique » disponible commercialement, pour répondre à ces besoins ?

Grâce aux services (gratuits ou payants) de TA en ligne, l'expérience des systèmes de TA n'est plus le privilège des experts. Mais le lecteur internaute moyen est souvent frustré par la pauvreté des résultats. En effet, le lecteur peut très facilement trouver des erreurs dans les phrases produites par la machine dans sa langue.

Peut-on espérer que les « traducteurs web » s'améliorent rapidement et deviennent utilisable pour de la communication multilingue de qualité ?

D'après [Hutchins 02], les changements principaux dans le domaine de traduction automatique depuis les années 90, sont dus aux facteurs suivants :

l'utilisation croissante de la TA par les grandes entreprises

l'exploitation des mémoires de traduction et d'autres outils constituant des poste de travail de traduction

les besoins croissants en localisation

la croissance de l'usage des ordinateurs personnels

l'impact d'Internet

la traduction en ligne

l'intégration de la TA et des autres activités de TALN (traitement automatique de langue naturelle)

la recherche de méthodes basées sur les corpus (TA statistique, TA par l'exemple), à mi-chemin entre les mémoires de traduction et la TA fondée sur des connaissances explicites (linguistiques et sémantiques).

Rien dans l'évolution indiquée ne permet d'espérer une augmentation significative de la qualité en domaine ouvert. L'architecture binaire de la plupart des systèmes garantit

aussi que la très grande majorité des couples de langues ne pourra pas être couverte, sauf par composition de deux systèmes, menant à une qualité encore plus faible. Il nous faut autre chose que la TAO actuelle.

1.2 Pour la TA multisource et multicible, une architecture à pivot interlingue est nécessaire

Pour créer et maintenir un document multilingue, en permettant d'augmenter incrémentalement sa qualité par partage du travail de révision, la meilleure approche nous semble être d'utiliser un « interlingua formel (IL) » et:

de répercuter les modifications d'une langue naturelle L_0 vers l'IL,

de régénérer vers les autres langues naturelles L1, … Ln depuis l'IL (L_0,..,L_n sont les langues naturelles dans le système).

Dans un système de traduction multilingue, si nous utilisons une structure pivot, le nombre des dictionnaires est 2 N, N étant le nombre des langues dans le système. Mais il faut aussi considérer que le coût de construire un dictionnaire pivot-LN est sans doute 3 fois plus élevé que celui de LN-LN [Boitet 90d]. Avec cette hypothèse, le coût principal d'un tel système est 3*2*N=6 N.

D'autre part, dans un système à transfert, l'idée reçue selon laquelle le nombre des composants serait quadratique n'est pas correcte. Supposons par exemple qu'on utilise comme « pivot non-interlingue », les structures-uma (unisolution, multiniveau et abstraites) d'une langue particulière. On peut réaliser toutes les traductions entre N langues avec 2N-2 transferts. Sur les N(N-1) couples, 2N-2 seront réalisés par transfert lexical simple et (N-1)(N-2) par transfert lexical double. Notons qu'il y a toujours double transfert lexical dans une approche à pivot « interlingue » [Boitet 88b].

Dans la pire architecture à transfert possible, avec N(N-1) transferts, si nous comparons le coût des composants de ce système à pivot (6N) et le coût d'un système de transfert (N(N-1)), le système à pivot est moins cher seulement quand N est plus grand ou égal à 8 (quand $N(N-1)-6N>0$)[1].

Cela dit, l'architecture à N(N-1) transferts est trop naïve et personne ne l'utilise. On prend plutôt les résultats d'analyse d'une langue par le système comme « langue pivot ». Dans ce cas, le coût principal d'un tel système sera 2(N-1). Mais dans la réalité, on n'a pas de très gros corpus ni assez de linguistes compétents sur la structure de surface de la langue pivot, surtout quand la couverture dépasse les langues bien dotées. Si on prend l'anglais (une classe de structures d'analyse de l'anglais) comme pivot, il faut des développeurs connaissant très bien l'anglais et une théorie linguistique de la structure syntaxique de l'anglais. Cela est infaisable pour beaucoup de langues.

---

[1] Nous pouvons voir aussi, au contraire de l'efficacité qu'on croit en la structure pivot, que même le coût de construction du dictionnaire d'un système à pivot peut être quadratique. Si à cause de la nature du lexique pivot (par exemple, des définitions comme dans le projet CICC), il faut regarder les équivalents possibles d'un mot à introduire en L1 et les symboles pivot correspondants. Le coût peut être : $C(N)= k_0N^2+k_1N+k_2$, avec k_0 petit.

En bref, quand il s'agit de la structure (intermédiaire) de surface d'une langue naturelle, par exemple un pivot syntaxique, le transfert sera très compliqué et on aura du mal à trouver des développeurs. C'est pour cela qu'on a besoin d'une « structure abstraite » la plus interlingue possible, et pas d'une « structure concrète » d'une langue particulière.

Enfin, la structure pivot est plus efficace quand il s'agit d'un système fortement multilingue (N _ N langues). En effet, il est plus facile d'ajouter une nouvelle langue dans un système à pivot interlingue, car il n'y a en principe pas de « transfert structural » à écrire, alors qu'il faut en écrire deux si on utilise un « pivot linguistique » comme les structures multiniveau de l'anglais.

1.3 Diminution des coûts par partage de la révision /post-édition en TA multilingue - l'idée de la coédition

Il est incontestable qu'on n'obtient de bons résultats en TAO qu'avec des systèmes à domaine fixé, à préédition ou entrée contrôlée, et/ou de type KBMT (knowledge-based machine translation). Mais nous visons d'autres contextes, et ne pouvons utiliser ce type d'approche. Nous visons en effet un système de domaine général et utilisable par l'utilisateur ordinaire. Or, on ne peut pas demander à un utilisateur ordinaire sans entraînement d'écrire en langage contrôlé. De plus, même dans le système CATALYST de CMU-Caterpillar à domaine fixé et à entrée contrôlée, et utilisant une ontologie, la postédition (révision) est toujours nécessaire pour obtenir un résultat précis [Hutchins 02]. Il nous semble donc que la postédition sera toujours indispensable pour obtenir une bonne ou très bonne qualité.

L'innovation majeure que nous apportons est un moyen de ne faire la révision qu'une seule fois et, dans une seule langue cible, pour chaque passage révisé (mais peut-être dans deux langues différentes pour deux passages différents), et d'en faire bénéficier automatiquement les autres langues cibles.

En quoi consiste au juste la post-édition de TA ? La post-édition n'avait pas été prévue dans les systèmes de TA du tout début, qui devaient remplacer le traducteur. On avait simplement oublié que, en traduction professionnelle, le travail du traducteur, même excellent, est toujours révisé par un « senior ». Dans la pratique, il existe comme on l'a dit des systèmes de TA assez spécialisés pour qu'on puisse utiliser leurs résultats comme des premiers jets de traducteurs humains et les soumettre à des réviseurs.

Dans la pratique, le temps pour la révision humaine d'un document issu de TA est environ un tiers de celui de la traduction humaine. Chaque page standard de 250 mots demande environ une heure pour la traduction. Prenons 30 pages standard de 250 mots. Pour traduire et réviser ces pages en N langues à la main, le temps estimé est (30+10)N=40N heures (traduction + révision). Dans un autre cas extrême où la TA du réviseur (TAO-R) est disponible, le temps demandé sera 10N (seulement le temps de révision). Bien sûr, le temps pour les autres moyens (THAM, par exemple) se situe au milieu. On a donc l'équation suivante : (pour 30 pages standard, soit 7500 mots, ou 42000 caractères) :

TAO-R(10N) < THAM < THum(40N)

Si on a une structure pivot sur laquelle on peut réviser à travers une langue naturelle, et si la modification peut ensuite se propager dans les autres langues par génération,

on n'a besoin de réviser qu'une seule fois, comme dans la Fig. A-1. On peut éliminer la variable N. Même si la révision prenait plus de temps dans cet environnement (peut-être à cause de l'environnement guidé, ou à cause du fait que le texte de surface est lié à la structure interne), par exemple, s'il augmente de 50% (une demi-heure soit 15 heures pour 30 pages), l'approche serait quand même très rentable, et cela d'autant plus qu'il y a beaucoup de langues dans le système (N grand).

Coédition (15) < TAO-R(10N) < THAM < THum(40N)

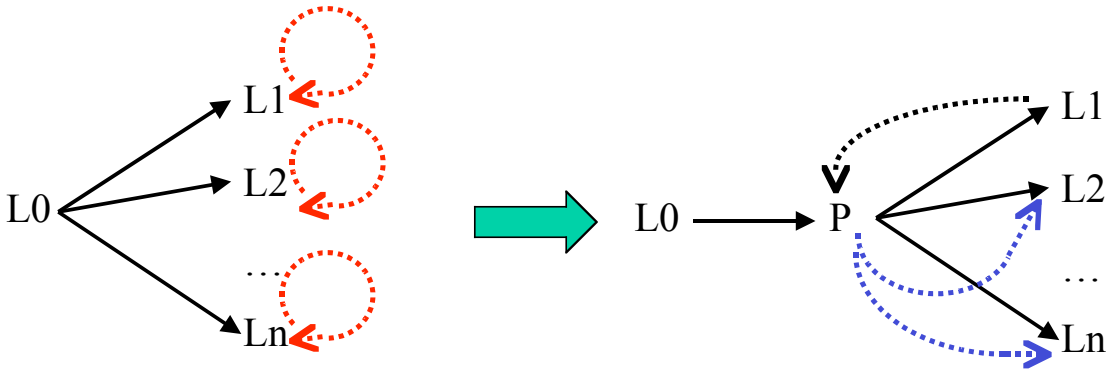


Fig. A-1 Partage de révision

Il faut noter que l'idée de « partager la révision » par coédition, ou autrement, est tout-à-fait nouvelle. Elle n'a pu émerger qu'à cause des progrès de la TA par pivot.

1.4 Utilisabilité par des non-spécialistes et des bénévoles

L'idée ici est que chacun peut être le réviseur ou le correcteur d'un document dans sa langue maternelle. Nous ne savons pas toujours pourquoi une phrase est incorrecte, mais nous avons toujours la capacité de donner une phrase similaire mais plus correcte. Dans un environnement bien guidé et contrôlé, tout un chacun devrait pouvoir utiliser des outils pour corriger un document dans la langue qu'il connaît. Notre idée est que la révision ne sera pas faite que par des professionnels, mais bien plus par les lecteurs eux-mêmes, et particulièrement sur les fragments jugés en valoir la peine.

D'où viendront ces « bénévoles de la coédition » ? Nous pensons qu'il y en aura, comme pour le développement de Linux, des outils du W3C et des shareware sur Internet. Les communautés internautes auxquelles nous pensons sont des groupes d'utilisateurs de produits grand public (matériels, logiciels, etc.) aussi bien que des groupes de discussion (Yahoo Clubs, MSN groupes, Lycos, Geocities, etc.). Peut-être arrivera-t-on donc aussi à motiver des internautes pour aider bénévolement à postéditer et coéditer.

Nous avons expliqué notre situation et les raisons pour lesquelles nous pensons utiliser la « coédition ». Nous allons maintenant examiner plusieurs systèmes de coédition/édition et leurs interactions avec l'utilisateur pour avoir une idée plus concrète sur le concept même de « coédition ».

2. Définition des notions principales concernant la coédition

Nous avons choisi sept systèmes de TA ou de génération automatique de langue naturelle. Le critère de choix est que ces systèmes doivent avoir deux objets à manipuler. Cela nous permettra de proposer une taxonomie des systèmes de coédition, puis de spécifier les caractéristiques désirables de notre système de coédition.

2.1 Présentation de quelques systèmes utiles pour préciser la notion de coédition

2.1.1 LIDIA (Large Internationalisation des Documents par Interaction avec l'Auteur)

Début 1990, la TAO (Traduction Automatique par Ordinateur) pour le rédacteur, ou « TAO personnelle » était un nouveau concept dont l'émergence avait été rendue possible tant par l'expérience acquise en « TAO lourde » (pour le veilleur ou pour le réviseur) que par l'évolution de la bureautique vers des outils très interactifs et multimédia (hypertextes) disponibles sur des postes de travail bon marché, connectables à des serveurs puissants.

Au lieu de réviser (postéditer) les traductions brutes produites en langue(s) cible(s), l'idée est de prééditer (indirectement) le texte source grâce à un dialogue du système avec l'auteur, dialogue visant tant à standardiser l'entrée (langage « guidé ») qu'à la clarifier (ambiguïté, ellipses, etc.). La structure profonde ainsi obtenue, étant correcte sur tous les plans (morphologique, sémantique, pragmatique), doit permettre de produire des traductions de grande qualité. La maquette LIDIA-I a été produite en 1994 au GETA pour valider ce concept [Blanchon 94].

L'architecture physique est un système distribué dans lequel les stations de rédaction (les machines Macintosh) communiquent avec un serveur de traduction. Le typage des unités à traduire, la correction orthographique, la standardisation terminologique, les mesures stylistiques et traitement des formules figées sont des tâches de la standardisation confiées à la station de rédaction. Les phases d'analyse, de transfert et de génération sont effectuées sur le serveur de traduction.

Blanchon a choisi de réaliser la station de rédaction comme une extension du logiciel de création d'hypertextes Hypercard, très largement disponible, à un coût très faible, et d'ores et déjà employé en documentation technique et industrielle (Renault, etc.) et en création personnelle multimédia. Dans un premier temps, un environnement de traduction de piles Hypercard vers le russe, l'anglais, et l'allemand a été créé. Le français était la seule langue source. Le serveur de traduction était un linguiciel de TAO multicible avec rétrotraductions (pour le contrôle) écrit dans l'environnement Ariane-G5 du GETA.

Voici un image de l'interface de démarrage sur la station de rédaction.

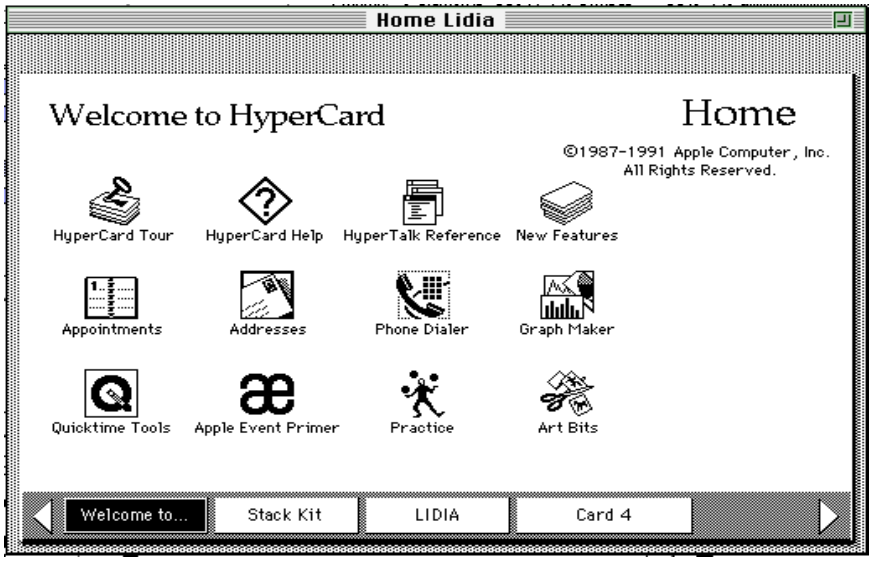


Fig. A-2 Interface (HyperCard) de démarrage de LIDIA-I

Les traitements principaux sont illustrés dans la figure A.2. Citons [Blanchon 94] :

1. Le texte français est d'abord standardisé sur la station de rédaction.
2. Le texte standardisé est alors analysé sur le serveur. La mmc-structure source produite (multisolution, multiniveau et concrète) est transformée en une forme portable (en lisp) et lisible (directement par les développeurs) et envoyée au Macintosh.
3. La mmc-structure source est utilisée pour produire le dialogue de désambiguïsation sur le Macintosh. Le processus de désambiguïsation la transforme en une umc-structure source non-ambiguë (unisolution, multiniveau et concrète) correspondant à l'analyse choisie par l'auteur.
4. Cette umc-structure source est alors « abstraite », ou « réduite » à une uma-structure source (unisolution, multiniveau et abstraite).
5. A partir de la uma-structure source, le système Ariane-G5 produit les gma-structures cibles (génératives, multiniveau, et abstraites), en utilisant les transferts adéquats. Une gma-structure est plus « générale » et plus « générative » qu'une uma-structure, car ses niveaux de surface (fonctions syntaxiques, catégories syntagmatiques, etc.) peuvent être vides, et sinon ne sont que des préférence indiquées par le transfert.
6. Pour chaque langue cible, la génération structurale produit à partir de la gma-structure cible une uma-structure cible homogène avec ce que serait le résultat de l'analyse (et de la désambiguïsation) du texte cible qui sera généré. Cette étape consiste à choisir la paraphrase à générer en calculant les niveaux de surface et à choisir une première approximation de l'ordre des mots à partir des niveaux plus profonds (relations logiques et sémantiques, traits sémantiques, etc.).
7. Le processus de traduction se termine par les générations syntaxique et morphologique. Quand tous les objets ont été traduits, on obtient la ou les piles images dans la ou les langues cibles.

8. Les uma-structures cibles peuvent être utilisées comme point de départ de rétrotraductions permettant à l'auteur (monolingue) de contrôler les traductions.

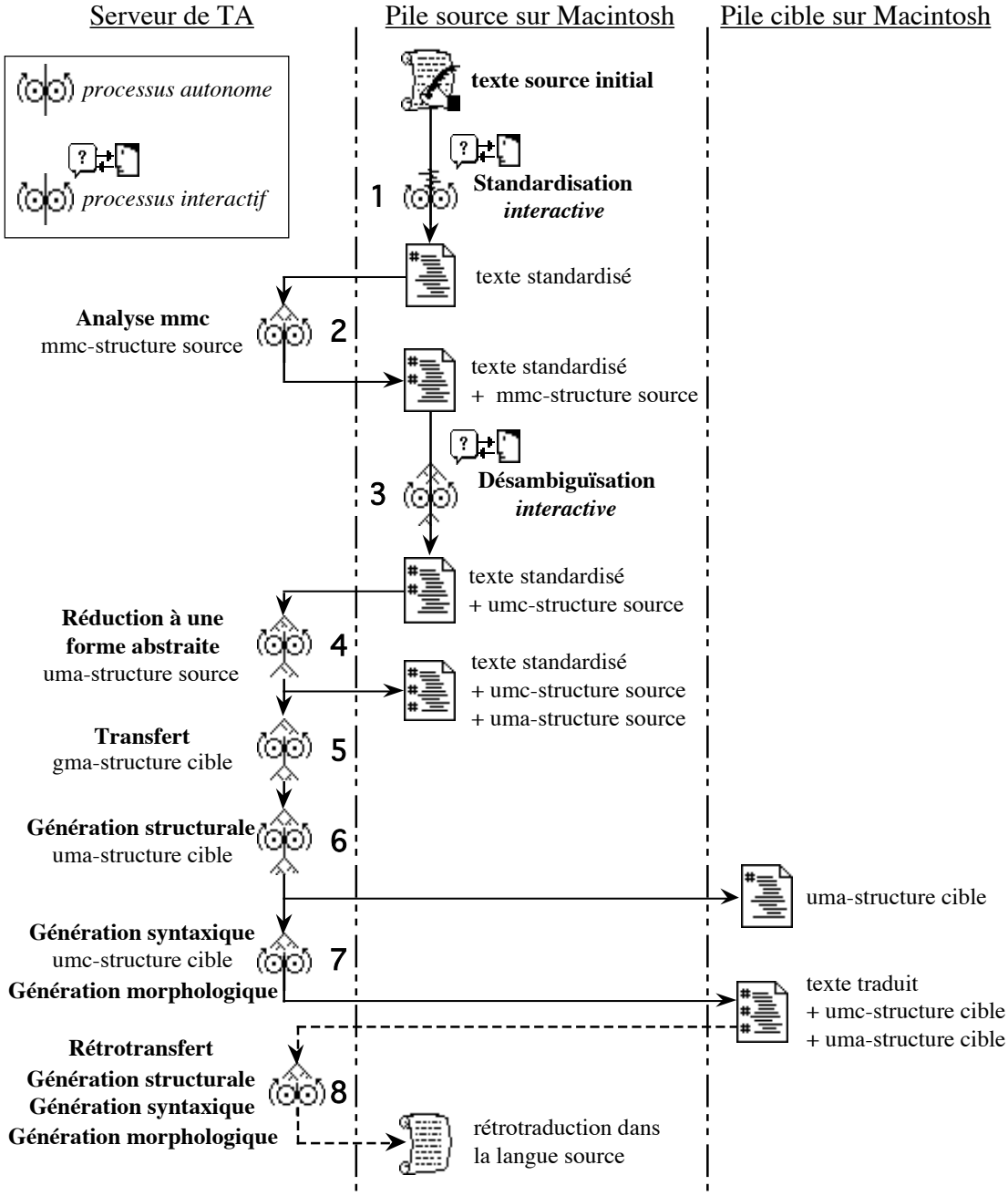


Fig. A-3 Organisation générale du processus de traduction en LIDIA-I

Le dialogue de désambiguïsation entre le système et l'utilisateur peut être sans ou avec explications selon le besoin et le niveau de l'utilisateur.

Voici une fenêtre de dialogue, sans explication. L'utilisateur peut cliquer sur le bouton pour demander plus d'explication.

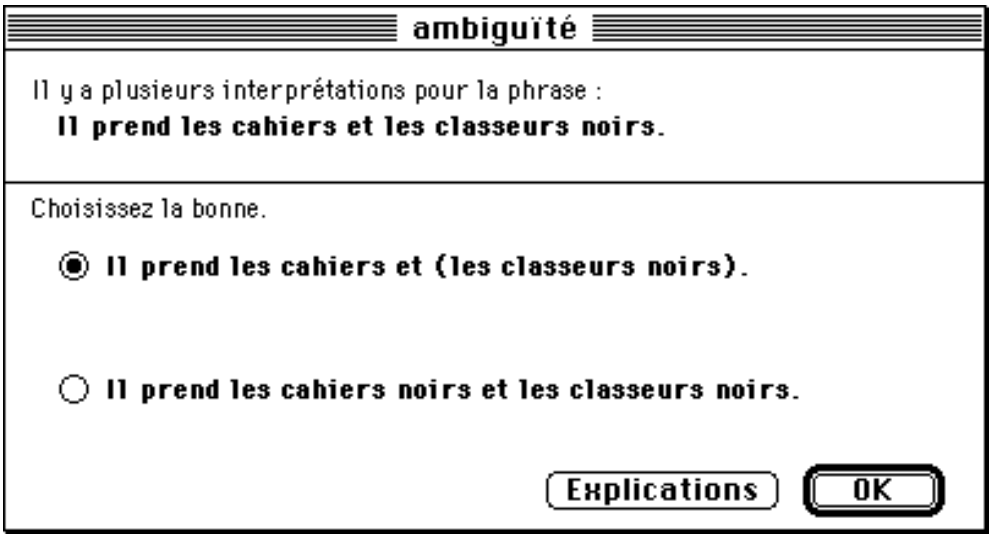


Fig. A-4 Dialogue avec paraphrasage et accès à des explications

Voici une figure qui montre la désambiguïsation avec explications. Quand l'utilisateur finit de lire l'explication, il peut retourner au dialogue et faire son choix.

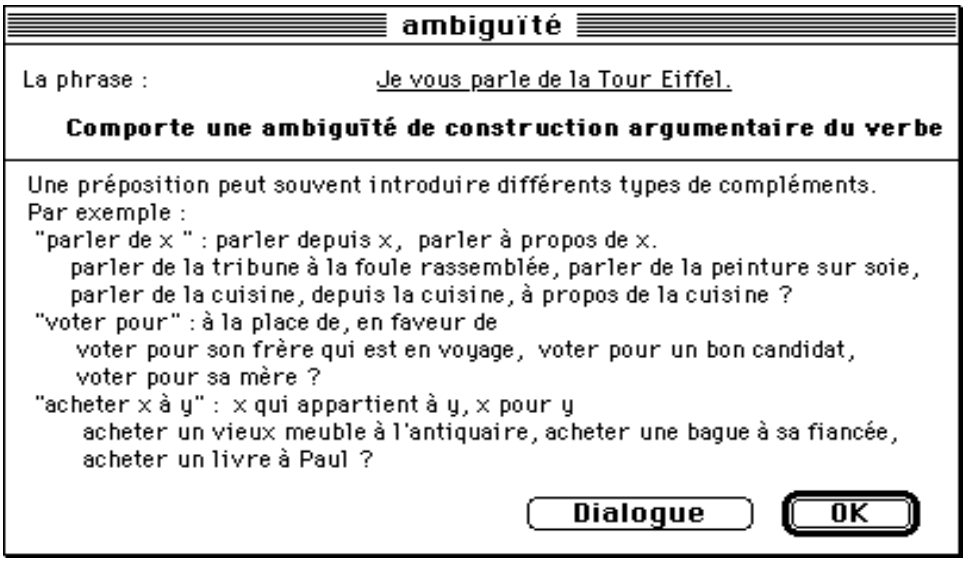


Fig. A-5 Explications pour l'ambiguïté de construction argumentaire du verbe

Analysons maintenant cette maquette pour dégager les aspects les plus pertinents de la notion de « coédition ».

2.1.1.1 Fiche d'identité

Objectif	Système de la TA et désambiguïsation interactive
Date	1994
Source ou description	Thèse de H. Blanchon "LIDIA-1: Une Première Maquette vers la TA Interactive pour TOUS"
Responsable	GETA, Hervé Blanchon
Langue source	français
Interface	Menu et fenêtre de dialogue
Langue	Français

d'interface	
Création de la structure interne	Après le parsage[2] de la phrase d'entrée, le système obtient plusieurs structures internes possibles, le système pose des questions à l'utilisateur en sa propre langue et l'utilisateur choisit la bonne structure
Structure interne	Arbres Ariane et données linguistiques
Langues cibles	russe, allemand, anglais (avec rétrotraduction en français)
Domaine	général
Site web	http://www-clips.imag.fr/geta/herve.blanchon
Utilisabilité	Tout le monde

2.1.1.2 Remarque

Bien que la structure interne et le texte de surface existent, LIDIA-I n'est pas un système de coédition, parce qu'il n'y a pas de modification en couple. Le processus de désambiguïsation interactive revient à choisir une structure parmi plusieurs structures possibles et l'utilisateur ne peut ni modifier la structure choisie ni les textes produits en différentes langues.

Il n'y a donc pas d'édition ni de coédition dans LIDIA-I.

2.1.2 MODEX

MODEX, développé par la compagnie CoGenTex Inc., est un produit de génération automatique de langue naturelle. L'intérêt de MODEX est qu'il montre dans un même projet 4 objets : le diagramme d'objets "OO" (object oriented), le plan du texte, le texte pour la validation du diagramme OO et le texte pour la documentation.

L'utilisateur prépare son texte par l'interface d'édition (planification du texte) et produit le diagramme OO comme représentation de connaissance.

Quand l'utilisateur veut vérifier le digramme OO, qui pourrait être difficile à comprendre, il peut demander de le sortir en texte (texte pour la vérification).

Dans le texte pour explication, l'utilisateur peut cliquer sur l'hypertexte (lié à une icône ou un identificateur de connaissance) pour voir plus d'explications sur ce mot, mais il ne peut pas éditer directement dessus. L'utilisateur est obligé de retourner à l'interface d'édition. Une fois satisfait, l'utilisateur peut produire le texte final.

Voici une vue du diagramme OO et une vue du texte pour la validation:

[2] Terme introduit par le linguiste québécois Jean-Yves Morin (Université de Montréal). Un analyseur accepte ou refuse une entrée, en produisant éventuellement une « image » de son processus d'acceptation, tandis qu'un parseur produit une structure définie indépendamment de l'histoire de l'acceptation.

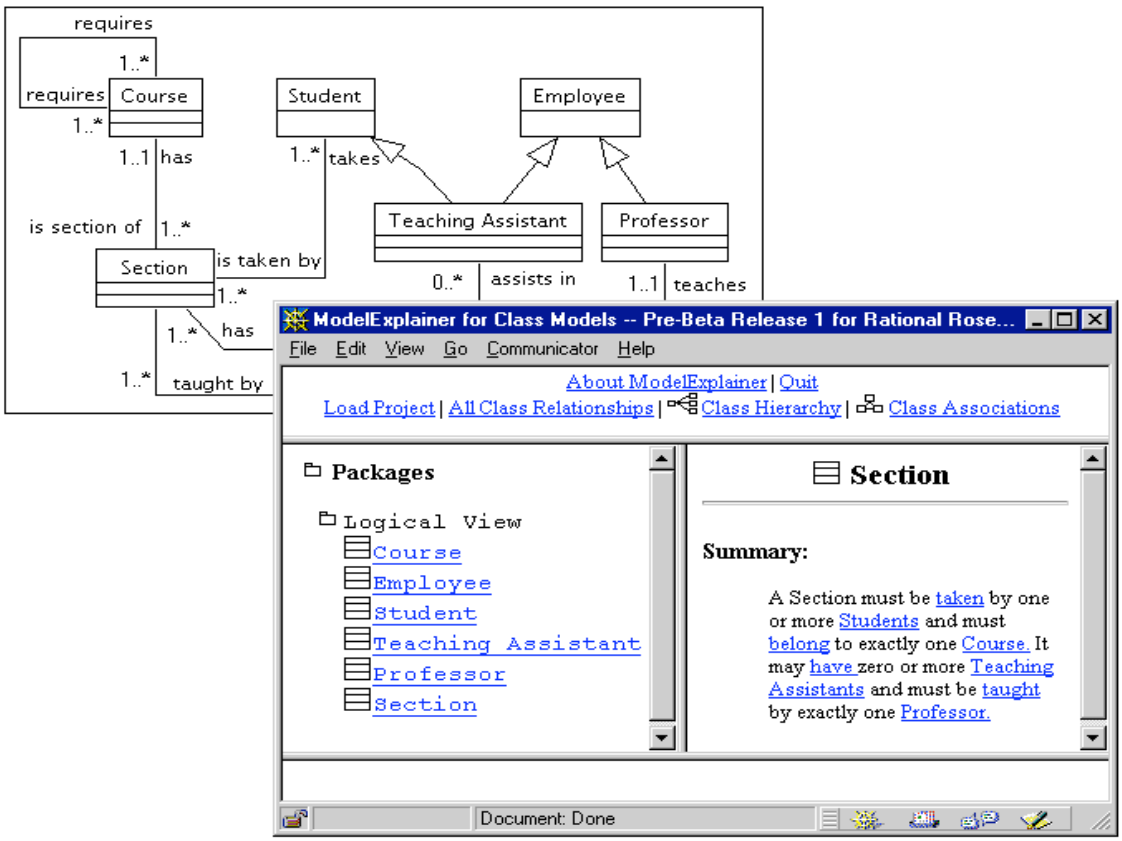


Fig. A-6 Image de MODEX

2.1.2.1 Fiche d'identité

Objectif	Produire des descriptions textuelles à partir d'un graphe d'objets. Les auteurs constatent qu'un diagramme d'objets est en fait plus difficile à lire qu'une description simple. Donc, ils ont besoin d'un système pour produire le texte explicatif.
Date	Proposé en 1997, maintenant commercialisé
Article	"Customizable Descriptions of Object-Oriented Models", Proceedings of the fifth Conference on Applied Natural Language Processing, Washington DC, pp. 265-268
Responsable	Lavoie, Benoit; Rambow Owen; Reiter Ehud
Interface	Il y a trois vues: le diagramme OO, la description pour validation, et le plan du texte.
Langue d'interface	anglais
Création de la structure interne	Manuellement avec l'aide de l'interface. Le système peut lire un diagramme OO puis y ajouter les données entrées par utilisateur sur ce diagramme
Structure interne	Modèle OO (structure sémantique, non syntaxique)
Langue cible	anglais
Domaine	Non-spécifique
Application sur autre domaine	possible, mais toujours domaine fixe

Utilisabilité	Expert du domaine
Site web	http://www.cogentex.com/research/modex/index.shtml

2.1.2.2 Remarque

Il n'y a pas de coédition dans le système MODEX. L'utilisateur ne peut que manipuler l'objet du plan de texte, et le système ne fournit aucun lien entre les autres objets. L'utilisateur ne peut pas voir le résultat de son édition tout de suite, il faut toujours attendre que le plan de texte soit terminé pour que le système puisse générer les autres objets.

2.1.3 DRAFTER

DRAFTER est un générateur destiné à produire des manuels multilingues de logiciel. Avec l'aide de l'interface, l'utilisateur crée la structure interne (objet O1) et finalement produit le texte de sortie (objet O2). L'intérêt de ce système est qu'il fournit à l'utilisateur la souplesse de définir ses propres classes de connaissances, en plus de ce qui est déjà défini dans la base de connaissances.

Voici l'interface de DRAFTER:

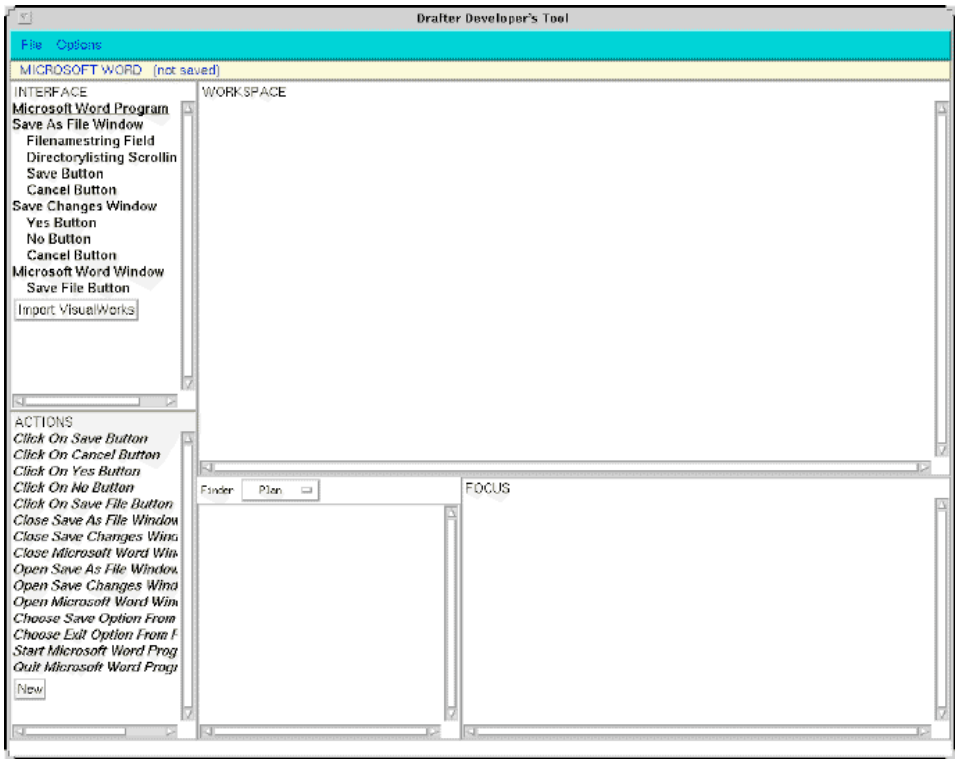


Fig. A-7 Interface de DRAFTER

2.1.3.1 Fiche d'identité

Objectif	Production de manuels multilingues de logiciels
Date	Mars, 1997
Source ou description	Hartley, A.F. et Paris, C (1997) "Multilingual document production: from support for translating to support for authoring", *Machine Translation, Special Issue on New*

	Tools for Human Translators, Vol. 12, no. 1-2, pp. 109-129
Responsable	ITRI
Interface	Menu, copier & coller objets
Langue d'interface	Anglais
Création de la structure interne	Manuellement avec l'aide de l'interface. L'utilisateur crée la structure interne en même temps qu'il édite l'interface graphique
Structure interne	Représentation conceptuelle
Langue cible	Anglais, français
Domaine	Manuels de logiciels
Application sur autre domaine	possible, mais toujours sur un domaine fixé et restreint
Utilisabilité	Expert du domaine
Site web	http://www.itri.bton.ac.uk/projectsindex.html
Remarque	Suivi par le projet AGILE (Automatic Generation of Instructions in Languages of Eastern Europe), qui s'étend au russe, au bulgare et au tchèque

2.1.3.2 Remarque

DRAFTER n'est pas un système de coédition, parce qu'une fois que le texte est créé, le processus est terminé. Nous ne pouvons pas prendre un texte et recommencer son édition ni faire de coédition.

2.1.4 Ambassador

Ambassador est un logiciel commercial qui a connu une grande réussite, mais ce n'est ni un système de traduction automatique ni un processeur de texte, C'est plutôt un système de traitement documents bilingues ou un système de coédition [Horn 95].

L'utilisateur choisit un patron de lettre. Deux lettres semi-finies s'ouvrent alors sur l'écran, l'une en français, l'autre en japonais. Le système permet à l'utilisateur de choisir dans les champs, avec des choix proposés par le système, ou d'entrer des données dans des zones libres. L'utilisateur peut choisir du côté français (objet O1) ou du côté japonais (objet O2) selon sa connaissance de chaque langue, et la modification faite se répercute tout de suite dans l'autre langue. Il existe aussi un petit dictionnaire dans le système et l'utilisateur peut ajouter de nouveaux mots.

Dans la Fig. A-8, nous voyons que l'utilisateur a tapé le nom du destinataire en français, mais il n'est pas encore affiché en japonais. Dans la même figure, nous pouvons constater qu'Ambassador ne traite pas la dépendance sémantique dans un document, à cause de l'inconsistance des sujets "je" et "nous" dans le document.

Ambassador vue I – Edition d'une lettre de "demande d'enquête"

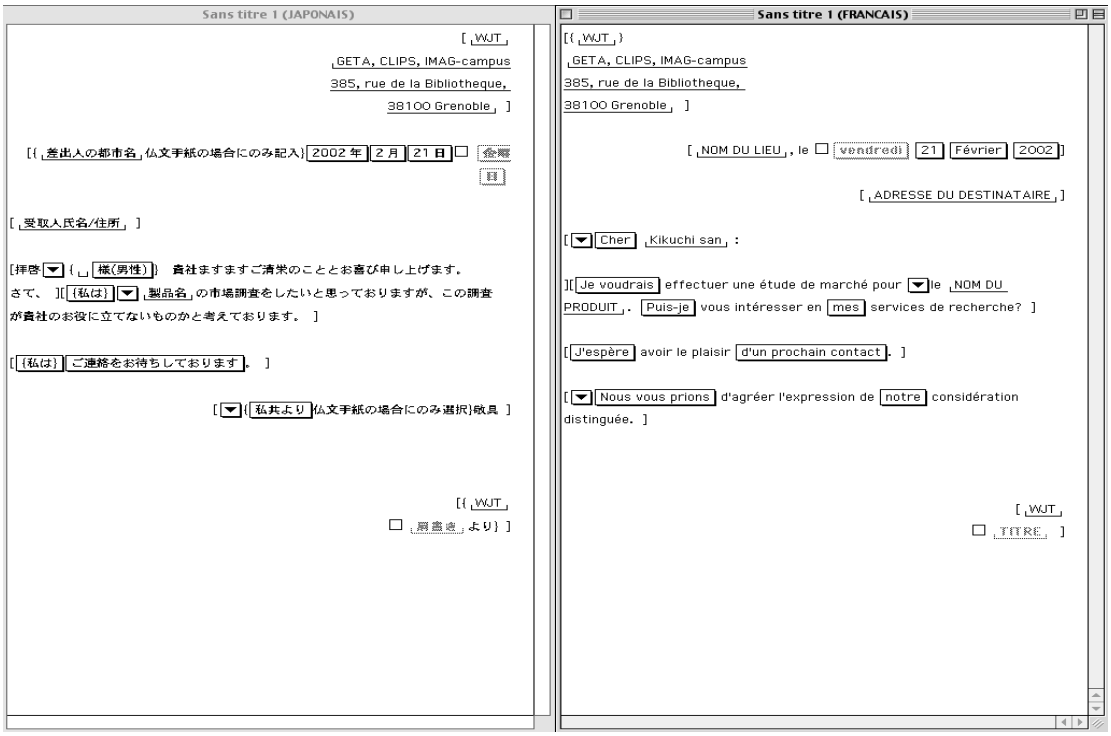


Fig. A-8 Ambassador vue I – Edition d'une lettre de « demande d'enquête »

Ambassador vue II – choix des phrases japonaises. Une fois que le choix est fait, la modification du côté français est immédiate.

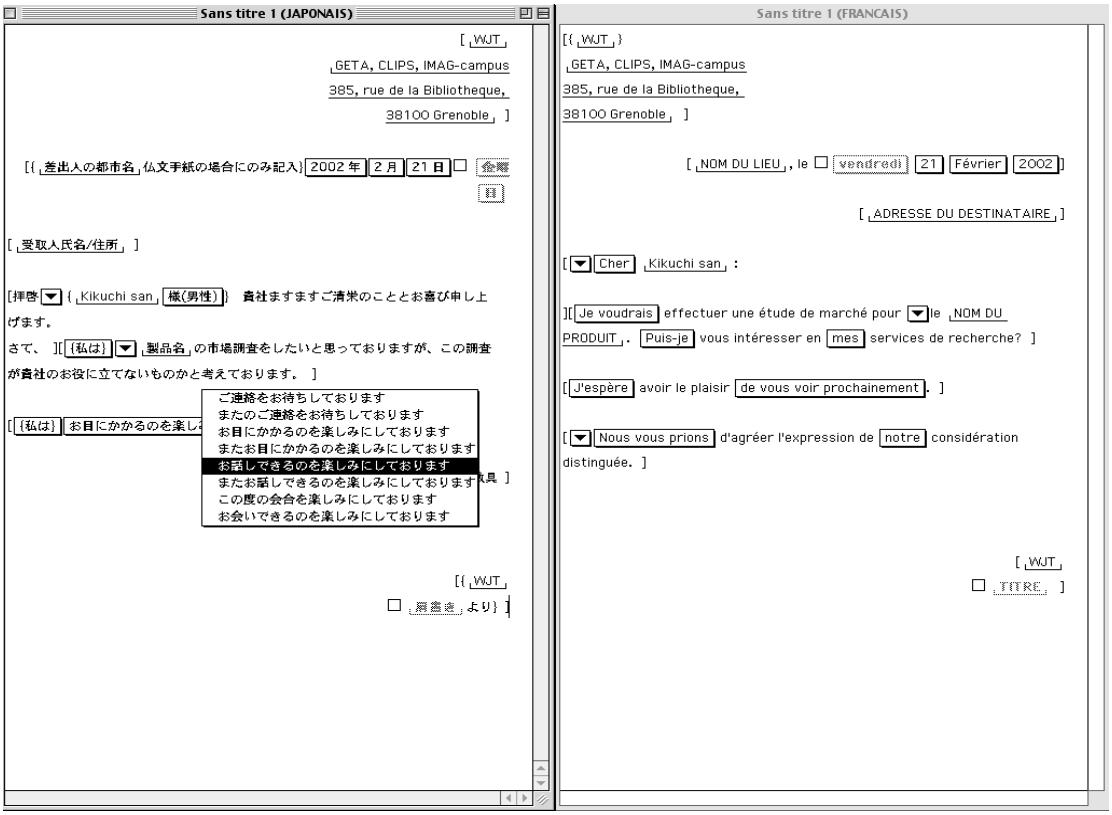


Fig. A-9 Ambassador vue II – choix au côté japonais

2.1.4.1 Fiche d'identité

Objectif	Système bilingue commercial pour produire les lettres d'affaires
Date	1995
Source ou description	Plus disponible
Responsable	Language Engineering Corporation, USA
Interface	Menu à choisir, et champs et zones libres à remplir
Langue d'interface	Anglais, japonais, français, espagnol
Création de la structure interne	Tout est encodé dans le système
Structure interne	invisible
Langue cible	Anglais, japonais, français, espagnol
Domaine	Production de lettres d'affaires
Application sur autre domaine	possible, mais toujours domaine fixe
Utilisabilité	Tout le monde
Site web	Pas disponible

2.1.4.2 Remarque

Nous n'avons pas trouvé d'information sur la structure interne d'Ambassador. Mais l'observation ci-dessus montre que ce système est fortement contrôlé en entrée. Le système n'a pas beaucoup de souplesse, mais il est très rapide et correct. Ambassador est un des systèmes de coédition les plus anciens.

Le fonctionnement de ce système nous mène à l'hypothèse suivante : il y a sans doute une structure interne qui se réduit à une table de correspondance. Un champ cliquable peut correspondre à plusieurs variables (choix possibles) et une variable peut apparaître dans plusieurs champs cliquables. Chaque variable établit une correspondance entre un segment de l'énoncé français et un segment de l'énoncé japonais.

L'utilisateur peut éditer l'objet O1 ou l'objet O2 dans Ambassador, et donc Ambassador est un système de coédition symétrique.

2.1.5 L'approche WYSIWYM (What you See Is What You Meant)

L'idée de l'approche WYSIWYM est que le système lie le texte de sortie et la structure interne. Donc, quand l'utilisateur édite le texte, il est en fait en train de créer ou d'éditer la structure interne. Nous disons que c'est de la coédition, parce que l'utilisateur édite un objet (la structure interne, objet O1) à partir d'un autre (le texte, objet O2).

Nous prenons comme exemple le premier système qui a utilisé l'idée WYSIWYM, DRAFTER II.

Pour créer un document, on remplit des textes cliquables en couleur proposés par le système. Le texte en rouge est nécessaire et le texte en vert est facultatif. Quand l'édition s'achève au bout de l'arbre de décision, le texte devient noir et il n'est plus

possible de le changer. Chaque fois que l'utilisateur clique sur un texte en couleur, le système lui propose des choix possibles dans ce contexte après avoir consulté sa base de connaissances.

Voici une image de l'interface WYSIWYM au début de l'édition d'un document. Il y a deux cadres, le texte d'édition en haut et la structure interne en bas. Il est aussi possible d'éditer directement la structure interne avec des actions limitées, par exemple, copier et coller.

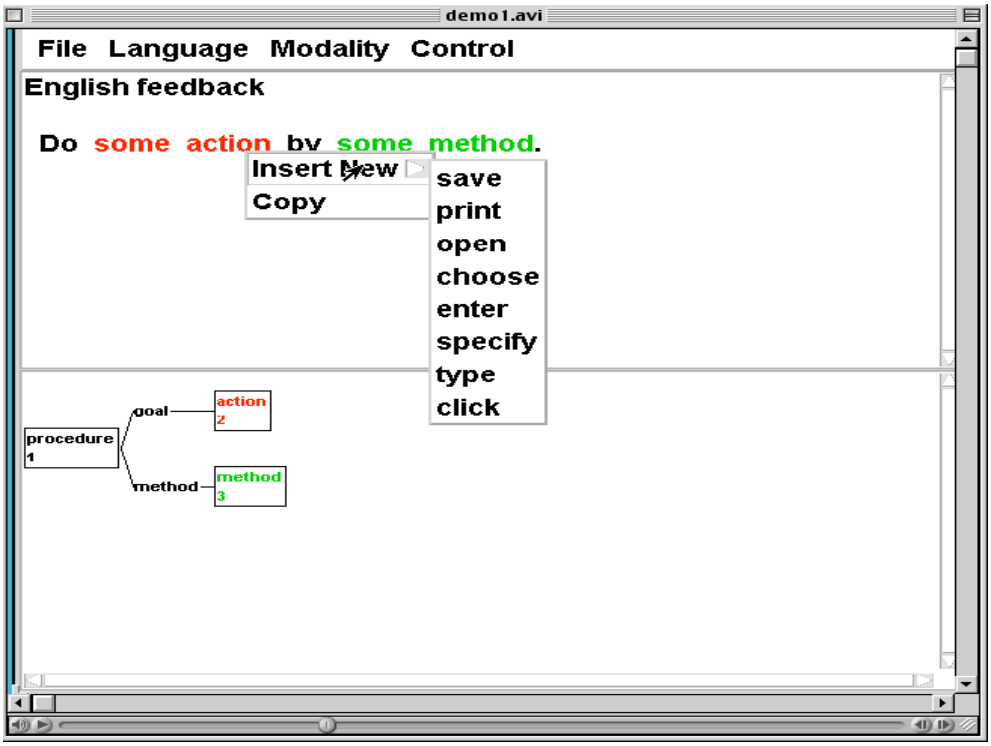


Fig. A-10 Début d'édition d'un document (système WYSIWYM)

Quand il n'y a plus de texte rouge, l'édition peut se terminer.

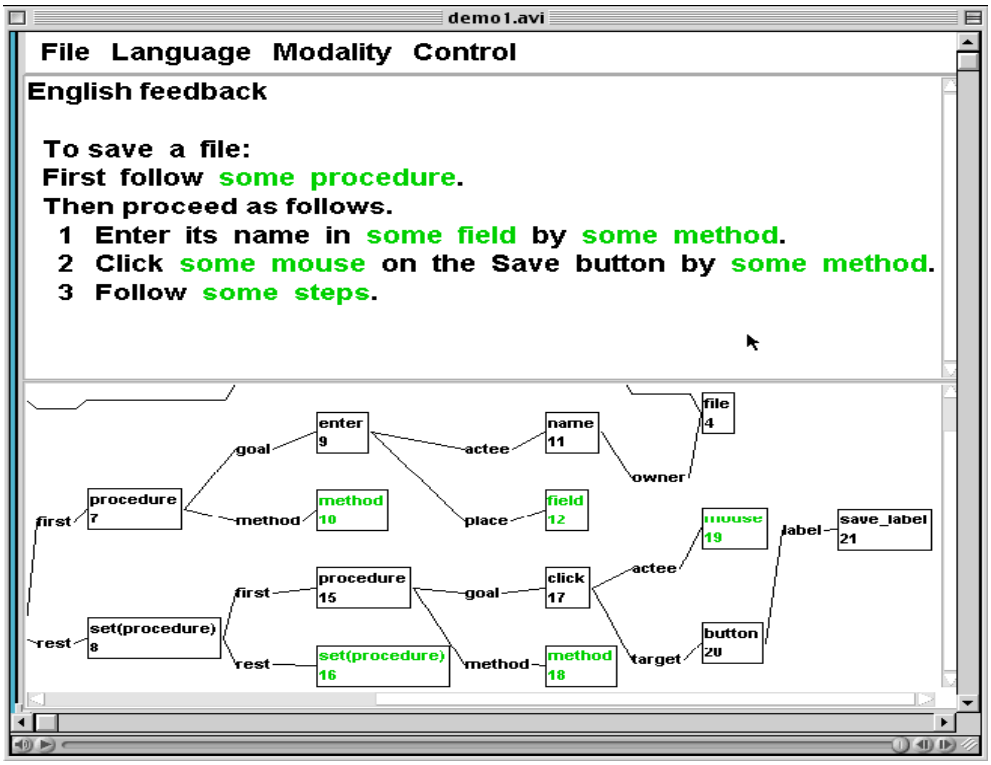


Fig. A-11 Fin d'édition d'un document (système WYSIWYM)

2.1.5.1 Fiche d'identité

Objectif	Produire les instructions pour utiliser un traitement de texte et un gestionnaire d'agenda
Date	1996-
Source ou description	R. Power, D. Scott and R. Evans (1998). What You See Is What You Meant: direct knowledge editing with natural language feedback. *Proceedings of the 13th Biennial European Conference on Artificial Intelligence (ECAI 98)*, Brighton, UK
Responsable	ITRI
Interface	Interface graphique avec menu et texte cliquable.
Langue d'interface	Italien, français, anglais
Structure interne	Représentation conceptuelle
Langue cible	Italien, français, anglais
Domaine	Production de manuels
Application sur autre domaine	Possible, mais toujours avec un domaine restreint
Utilisabilité	Utilisateur ordinaire
Site web	http://www.itri.bton.ac.uk/projectsindex.html
Remarque	A part DRAFTER-II, il existe aussi d'autres systèmes qui emploient l'idée de WYSIWYM, comme PILLS, CLIME, et ICONOCLAST. Toutes les informations sur ces systèmes se trouvent sur le site d'ITRI ci-dessus.

2.1.5.2 Remarque

L'idée de WYSIWYM est un bon exemple de coédition dans le sens texte _ structure interne. Le texte est traité comme l'interface d'édition de la structure interne. Donc, quand l'utilisateur édite la structure interne (objet O2), il l'édite en fait à travers le texte (objet O1).

Le point fort de WYSIWYM est que l'utilisateur édite directement le texte généré, donc il voit bien le résultat. Dans les systèmes précédents, l'utilisateur était obligé d'éditer une représentation à base d'icônes et de graphes, qui ne sont pas proches de langue naturelle, et donc ce n'était pas utilisable par tout le monde.

Bien sûr, derrière cette interface textuelle, il existe une structure représentative de connaissances.

Enfin, l'utilisateur peut aussi changer la langue de travail à tout moment, et obtient les autres versions, car le système a un générateur multilingue assez puissant.

Après DRAFTER II, l'idée de WYSIWYM a aussi été appliquée dans les projets PILLS [Bouayad-Agha 02], CLIME [CLIME] , et ICONOCLAST [ICONOCLAST].

Le système PILLS produit des descriptions de médicaments et le système CLIME produit de la documentation judiciaire.

ICONOCLAST (Integrating Constraints in Layout and Style) est un projet qui intègre la génération automatique de langue naturelle et les contraintes de style et de mise en forme. Avec ICONOCLAST, on peut sauvegarder un document et le rééditer plus tard, ce qui est un progrès par rapport aux systèmes DRAFTER, PILLS et CLIMS, qui sont plutôt destinés à la création de documents.

2.1.6 Multimeteo

Dans Multimétéo, le système produit un bulletin météorologique semi-fini à partir des données (il existe déjà plusieurs systèmes pour la génération automatique de bulletins météorologiques multilingues, comme FoG, MLWFA, etc.).

L'utilisateur affine ce bulletin brut *a posteriori* en cliquant sur le texte en rouge. Le système lui propose alors des choix possibles. L'utilisateur édite donc un *interlingua* à travers le GUI (Graphical User Interface).

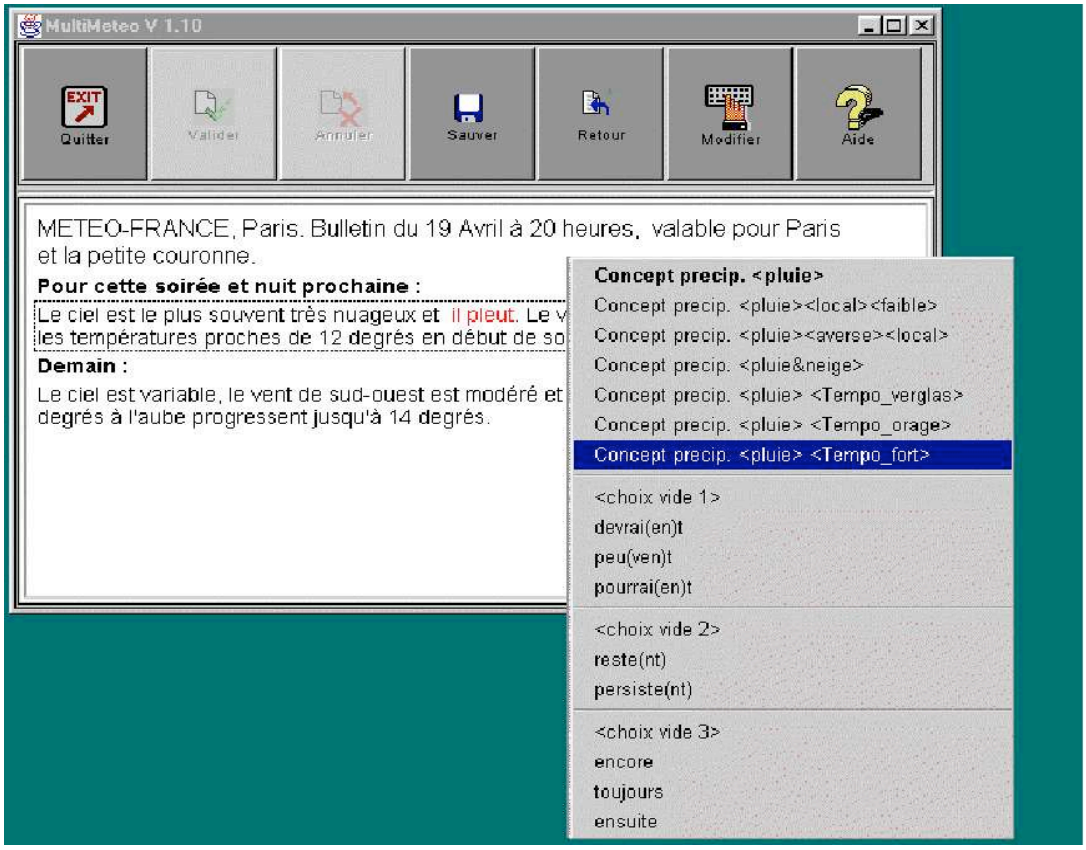


Fig. A-12 Interface de Multimétéo

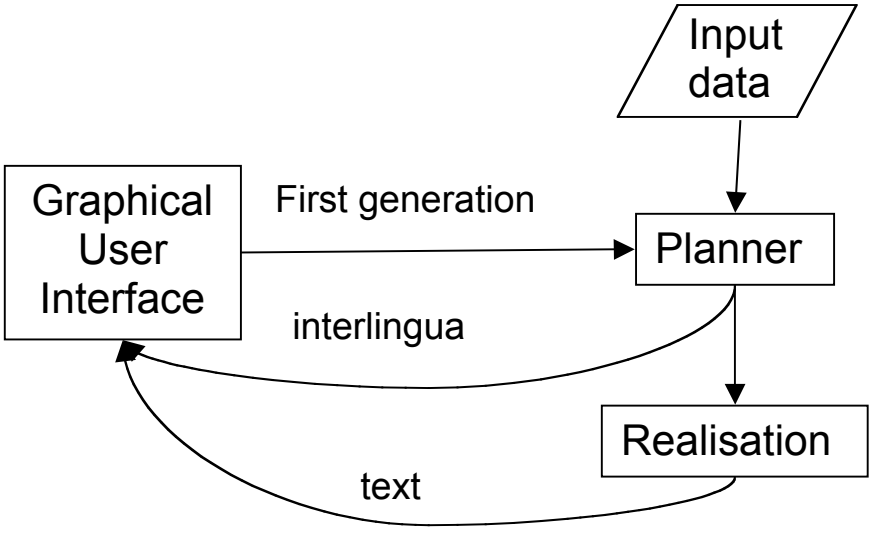


Fig. A-13 Procédure d'édition du système Multimétéo

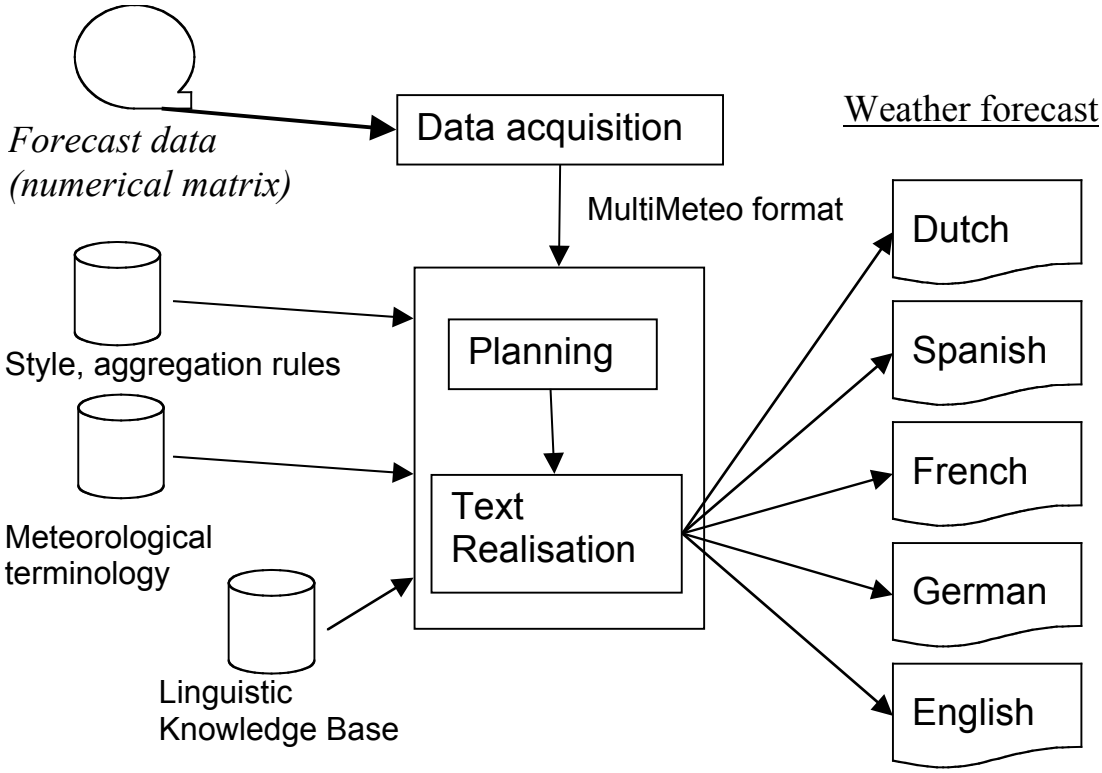


Fig. A-14 Structure générale du système Multimétéo

2.1.6.1 Fiche d'identité

Objectif	Produire interactivement un document multilingue de prévision météorologique
Date	Multimétéo II 1999-
Source ou description	(2001) José Coch, Karine Chevreau, "Interactive Multilingual Generation" CICLing-2001 (Computational Linguistics and Intelligent Text Processing), Mexico, February 2001
Responsable	Météo-France, INM, ZAMG, IRM & LexiQuest
Interface	Menu et description textuelle avec des mots cliquables.
Langue d'interface	Anglais, espagnol, français, allemand (néerlandais, catalan, galicien, basque à venir)
Structure interne	Représentation conceptuelle
Langue cible	Anglais, espagnol, français, allemand (néerlandais, catalan, galicien, basque à venir)
Domaine	Bulletin météorologique
Application sur autre domaine	Possible, mais toujours sur un domaine restreint
Utilisabilité	Interface facile à utiliser par un utilisateur ordinaire
Site web	http://www.knmi.nl/hirlam/NewsLetters/35/OperationalEs/mmeteo/Multimeteo.html
Remarque	

2.1.6.2 Remarque

Multimeteo est aussi un bon exemple de coédition. L'utilisateur édite directement le texte (objet O1) et la modification est faite sur la structure interne (objet O2) et le texte (objet O1) lui-même, comme pour WYSIWYM. C'est pourquoi nous appelons ce genre de coédition « coédition double ».

2.1.7 MDA (Multilingual Document Authoring)

MDA est un système de composition de documents multilingues qui a été appliqué à la production de modes d'emploi de médicaments. L'interface est une sorte d'union de celle d'Ambassador et de celle de WYSIWYM :

La mise en forme du document est décidée au début comme avec Ambassador.

La structure de document peut évoluer au cours de l'édition comme avec WYSIWYM, mais avec des choix plus sophistiqués et aussi une dépendance syntaxique plus stricte.

La structure interne est aussi différente (DTD enrichie).

Voici une image d'interface de MDA :

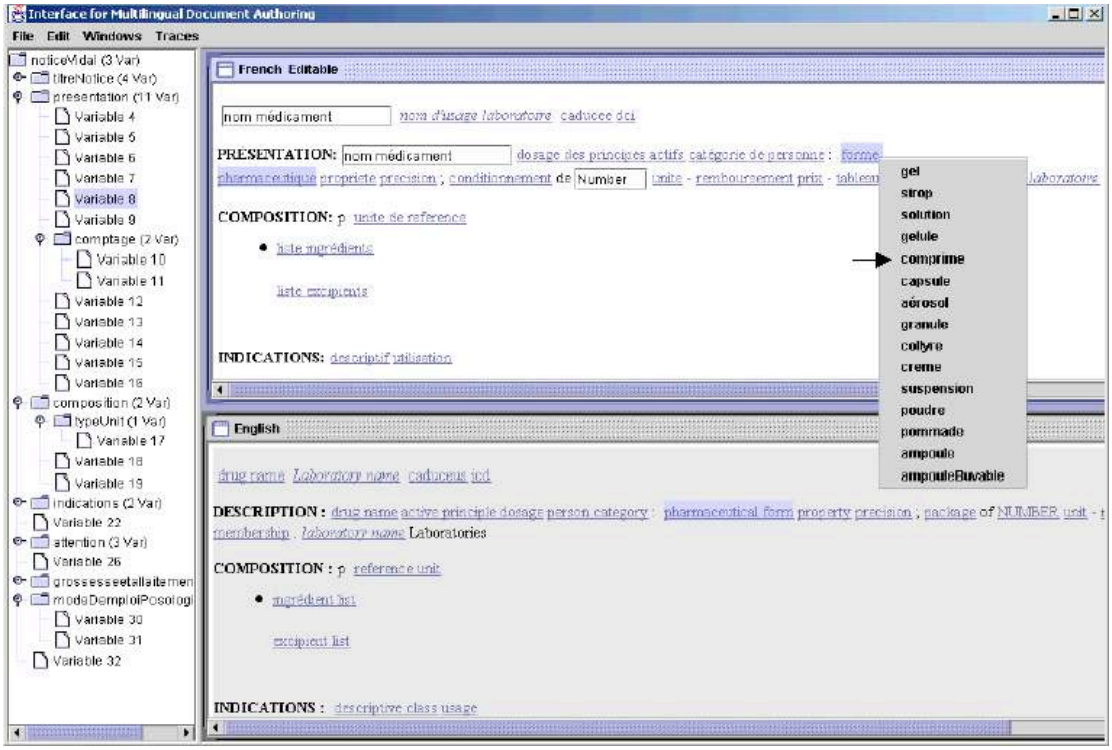


Fig. A-15 Interface de MDA

2.1.7.1 Fiche d'identité

Objectif	Produire des modes d'emploi multilingues de médicaments
Date	2002-
Source ou description	"Document Structure and Multilingual Authoring", Caroline Brun, Marc Dymetman, Veronika Lux, *Proc*

	INLG'2000, Mitzpe Ramon, Israël, pp 24-31
Responsable	XRCE
Interface	menu
Langue d'interface	Anglais, français, allemand
Structure interne	DTD enrichie
Langue cible	Anglais, français, allemand
Domaine	Modes d'emploi multilingues de médicaments
Application sur autre domaine	Possible, mais toujours sur un domaine restreint
Site web	http://www.xrce.xerox.com/competencies/content-analysis/dcm/demo/mda-demo.html
Utilisabilité	Utilisateur ordinaire
Remarque	Mise en relief d'un document sémantiquement correct

2.1.7.2 Remarque

Nous remarquons qu'il y a aussi deux cadres dans l'interface. Mais, au lieu de montrer deux structures comme WYSIWYM, MDA montre le texte en français et en anglais dans deux cadres.

On ne peut pas éditer directement la structure interne, mais cela n'empêche pas MDA d'être un bel exemple de coédition double.

MDA offre aussi le moyen d'exprimer des contraintes sémantiques très fortes, par exemple, la compatibilité sémantique entre les champs remplis.

2.2 Aspects principaux

L'étude des systèmes précédents nous a amené à dégager quelques concepts utiles, dont nous tentons maintenant de donner une définition précise.

2.2.1 Définitions

Nous nous plaçons dans la situation où on veut modifier deux ou plusieurs objets fortement reliés entre eux, comme un texte et sa (ou ses) structure(s) abstraite(s). Il ne s'agit pas d'éditer un objet unique à travers plusieurs vues sémantiquement équivalentes (comme par exemple les vues « normales », « page », et « plan » de Word). En effet,

> une même structure interne peut en général correspondre à un nombre variable de textes (paraphrases) plus ou moins exactement synonymes,
>
> un même texte peut aussi correspondre à des structures internes différentes et d'interprétations différentes (pas d'ambiguïté).

Nous prenons le cas le plus simple, où il n'existe que deux objets et nous appelons ces deux objets « O1 » et « O2 » dans les définitions suivantes :

Définitions

coédition - édition de O2 à travers O1.

édition - modification de O1 ou O2 par un série de modifications locales.

localité - portée d'une modification (normalement, inférieure à l'énoncé).

coédition double - édition de O2 *et* de O1 à travers O1.

coédition simple - édition de O2 *et pas* de O1 à travers O1.

coédition pseudo-double - édition de O2 *et pas* de O1, à travers O1, puis mise à jour produisant (sur demande) O1' sous une forme montrant les différences entre O1 et O1' de façon analogue à *ce qu'aurait pu produire une édition de O1*.

coédition symétrique – on peut coéditer O1 à travers O2 de la même manière qu'on coédite O2 à travers O1.

coédition contrainte/libre – l'utilisateur ne peut éditer que certaines parties du document (dans O2), ou l'utilisateur peut éditer n'importe quelle partie du document (dans O2).

génération immédiate - une édition sur O1 se propage immédiatement à O2 (ici en général, O1 est la structure interne).

Tableau A-1 Taxonomie de la coédition

2.2.2 Application de cette taxonomie aux systèmes étudiés

Voici un tableau résumant les définitions, et le type de coédition offert par les systèmes présentés ci-dessus :

	Nature et opération		Coédition simple			Coédition pseudo-double (si pas de coédition double)
	Objet 1	Objet 2	simple	double	symét-rique	
LIDIA	Ensemble d'arbres à sélectionner	Texte génération totale	non	non	non	non
Ambassador	Texte d'édition contrainte	Texte d'édition contrainte	oui	oui	oui	
MODEX	Texte/objet d'édition libre	Texte génération totale	non	non	non	non
DRAFTER	Objet d'édition contrainte	Texte génération totale	non	non	non	non

WYSIWYM	Texte d'édition contrainte	Structure interne	oui	oui	non	
Multimeteo	Texte d'édition contrainte	Structure interne	oui	oui	non	
MDA	Texte d'édition contrainte	DTD enrichie	oui	oui	non	

Tableau A-2 Taxonomie des systèmes étudiés

2.2.3 Comparaison synthétique

Une brève comparaison des systèmes de coédition précédents sera utile pour définir l'architecture adaptée à nos besoins.

Ambassador est le moins souple, mais il est précis et rapide, et facile à utiliser. En effet, toutes les correspondances possibles entre le texte et la structure interne sont établies a priori.

DRAFTER II (WYSIWYM) est également rapide et facile à utiliser, mais il est contraint par la nécessité d'une base de connaissances. Un autre défaut est que le texte sorti manque un peu de style. Ce défaut est amélioré dans le système ICONOCLAST. Le feedback textuel peut paraître peu naturel, par exemple : « do some action by some method ».

Multimeteo et **MDA** peuvent produire des textes assez bons mais ne sont applicables qu'à un domaine fixé. L'interface de coédition est presque en langue naturelle, donc ils sont assez faciles à utiliser.

Tous ces systèmes visent un domaine fixe. Ainsi, les correspondances entre le texte et la structure interne peuvent être encodées à l'avance. C'est grâce à cette restriction qu'on peut obtenir un résultat assez satisfaisant, mais c'est aussi à cause de cette restriction qu'il est impossible d'étendre ces systèmes au domaine général.

Nous constatons aussi qu'aucun des systèmes vus ici ne permet aux utilisateurs d'entrer du texte libre. Les utilisateurs ne peuvent que choisir parmi les choix proposés par les systèmes. En effet, dans ce type d'interaction homme-machine, il est sans doute impossible pour la machine de comprendre ce que veut dire l'homme, et il est aussi plus efficace pour la machine de proposer des modifications possibles selon la structure interne et le contexte, au lieu de comparer pour trouver la correction faite par l'homme. Donc, nous pouvons pour l'instant supposer que ce genre d'interaction (l'homme choisit, la machine propose) sera utilisé dans tous les systèmes.

2.3 Types de coédition souhaitables

Nous nous situons toujours dans la perspective d'un système à large couverture, non restreint à un domaine et un type de documents particuliers, et utilisable par le grand public (en mode non expert). :

coédition – Nous souhaitons un système de coédition, surtout dans le sens texte _ structure interne. Pour l'utilisateur ordinaire, le texte est en effet le moyen le plus naturel de s'exprimer et d'éditer (ou d'annoter).

coédition double – La coédition double est souhaitable, parce que plus rapide, mais elle n'est pas indispensable. En effet, l'objet (O1) au travers duquel on édite l'autre objet (O2) sera régénéré à l'étape suivante à partir de la forme interne. D'autre part, système de coédition double est plutôt difficile à construire, sauf quand toutes les correspondances sont prévues.

coédition symétrique – Un système de coédition symétrique n'est pas indispensable, parce que, pour la plupart des utilisateurs, la structure interne reste difficile à comprendre, et il est donc presque impossible de l'éditer directement. Le seul intérêt de la coédition symétrique est pour les experts. De plus, un système de coédition symétrique est encore plus difficile à construire si les deux objets sont hétérogènes, ce qui est notre cas.

coédition pseudo double – Ce genre de système peut être intéressant pour simplement montrer les traces de correction, mais cette fonctionnalité ne paraît pas indispensable.

coédition libre / contrainte – Nous souhaitons que l'utilisateur de notre système ait la liberté de modifier n'importe où dans le document. La portée de toutes les modifications est alors locale, et donc l'édition est plus simple et légère pour l'utilisateur ordinaire.

domaine – Un objectif essentiel est que notre système puisse traiter le domaine général. Il est alors sans doute impossible de prévoir toutes les correspondances et les connaissances. C'est pourquoi nous nous proposons de construire les correspondances a posteriori pour faire la coédition.

En bref, un système de coédition idéal serait un système qui s'appliquerait au domaine général, et qui permettrait la coédition double, symétrique et libre.

3. Comment adapter l'idée de coédition à la communication multilingue écrite/orale

3.1 Architecture linguistique générale "à pivot"

3.1.1 Utilisation d'une représentation interlingue pivot

Puisque nous voulons produire un système multilingue, une représentation pivot (intermédiaire) est plus commode et souple pour ajouter une autre langue dans le système. Pour un système de coédition, selon notre discussion précédente, il n'est probablement pas possible de coéditer deux langues naturelles comme deux objets de coédition, car cela est trop compliqué. Il faut donc avoir un pivot servant de base d'édition.

3.1.2 Production automatique ou semi-manuelle du pivot

Quelle que soit la représentation intermédiaire que nous utiliserons comme « pivot », ce sera une représentation abstraite d'un ou de plusieurs énoncés, difficile à lire et donc cachée aux utilisateurs non experts.

Dans notre système, pour les utilisateurs qui ne connaîtront pas cette représentation pivot, il devra y avoir des modules qui feront le transfert de LN vers le pivot automatiquement. Bien sûr, la qualité d'une représentation de pivot produite automatiquement ne pourra pas être parfaite, ce qui rendra la postédition (par coédition !) très utile, voire indispensable.

D'un autre côté, il faut garder la souplesse et permettre aux experts de produire ce pivot semi-manuellement pour que la représentation pivot soit la meilleure possible. Cela va de techniques d'analyse multiple suivie de désambiguïsation interactive à la construction manuelle assistée pas un environnement adéquat (manipulation directe de la structure, vérifications automatiques de cohérence).

3.1.3 Coédition séparée/indépendante des langues analysées

On souhaite pouvoir coéditer depuis toutes les langues dans le système, pas seulement depuis une langue spécifique. Il faut donc que le pivot ne soit pas lié à une seule langue, mais soit réellement interlingue.

3.2 Insertion dans des systèmes d'information

3.2.1 Aspect décentralisé

Pour inclure autant d'utilisateurs que possible, le système doit être décentralisé. En effet, nous ne pouvons pas construire un système centralisé qui inclut toutes les langues possibles.

Le système doit lui-même se présenter comme un module qui aide à la traduction. Ce module et donc ce module, par exemple, une applet Java, devra donc être facile à télécharger et être utilisable sur des plates-formes différentes.

3.2.2 Traitement local avec ressources minimales

On ne peut pas supposer que les utilisateurs disposeront d'analyseurs et de générateurs complets au moment de la coédition.

Il faudra donc que le système de coédition puisse fonctionner en n'utilisant que des ressources disponibles partout, comme

de simples dictionnaires LN-anglais,

des analyseurs morpho-syntaxiques.

3.2.3 Disponibilité sur Internet et Intranet

Le but final est de mettre notre système mise sur le réseau, Internet ou Intranet, pour qu'il soit ouvert à tout le monde. Bien entendu, la capacité de communication sur le

réseau et la programmation du réseau seront très importantes dans notre conception du système.

En particulier, il faut implémenter une technique simple permettant à plusieurs utilisateurs d'améliorer le même document en même temps, sans créer de conflits.

D'où l'idée de ne jamais rien effacer dans le document maître, mais d'y enregistrer les modifications de chacun comme des versions (monotonie).

3.3 Ingrédients d'une solution à pivot du point de vue des systèmes d'information

3.3.1 Un document maître XML-isé

Notre premier « ingrédient » sera donc l'utilisation d'un unique fichier multilingue en XML, ou « document maître XML-isé ». Ce document contiendra, pour chaque phrase, une ou plusieurs versions dans chaque langue, et une ou plusieurs versions sous forme pivot. Ce document maître sera bien sûr en Unicode. A partir d'un tel document, on pourra facilement produire des vues monolingues, ainsi que des documents monolingues (ou multilingues alignés) dans différents formats et codages.

3.3.2 Passage aisé entre deux modes de coédition - naïf et professionnel

Puisque notre système est destiné non seulement à l'utilisateur ordinaire mais aussi à l'expert, ces deux modes d'exploitation seront disponibles à tout instant et le passage entre ces deux modes devra être facile, pour encourager et attirer plus d'utilisateurs à participer et à entrer dans les détails de problème.

Bien entendu, le passage entre l'activité de lecture et celle de coédition doit lui aussi être rapide et non perturbant.

3.3.3 Choix de correction proposé par le système

Nous ne voulons pas laisser l'utilisateur éditer directement le texte, car ses modifications pourraient être incomplètes (par exemple, mise au pluriel d'un sujet et oubli de le faire pour le verbe) et surtout car il est très difficile d'interpréter des modifications textuelles à un niveau abstrait sans réanalyse totale. Mais c'est ce que nous voulons justement éviter !

D'où l'idée de proposer à l'utilisateur les modifications possibles sur la forme pivot, en les associant aux éléments correspondants du texte. Ainsi, si le curseur passe sur un mot, le système devrait proposer les modifications des parties correspondantes de la forme pivot, en les présentant comme des corrections à l'intention d'un typographe, c'est-à-dire comme des indications de « ce qu'il faudrait faire » (e.g., « mettre au pluriel »).

3.3.4 Établissement a posteriori des correspondances

Dans notre analyse sur les systèmes de coédition, nous avons vu que, pour un système de domaine général, on ne peut pas garder les correspondances tout le temps, parce que c'est trop compliqué. Par contre, les correspondances sont gardées dans les

systèmes contrôlés, à domaine restreint, parce que dans ce cas, la syntaxe et le vocabulaire sont limités et gérables.

Il nous semble donc qu'établir les correspondances a posteriori après chaque génération du texte est viable pour un système du domaine général.

3.3.5 Intégration de ressources gratuites

Comme il est très coûteux de construire des ressources linguistiques, nous concevrons notre système de façon à utiliser les ressources gratuites disponibles sur le web.

Nous limitons donc les ressources utilisables à

des dictionnaires et lexiques monolingues,

des dictionnaires bilingues, l'autre langue étant presque toujours l'anglais,

des segmenteurs et/ou baliseurs (tagger),

des analyseurs morphologiques.

Tous ces outils peuvent être accédés par le web où on peut faire une requête et recevoir le résultat par un CGI. Il est aussi souvent possible de télécharger une copie de ces programmes et les faire tourner localement, voire même d'accéder à leur source et de les modifier pour les intégrer dans un autre système.

A titre d'exemple, nous présentons maintenant trois outils gratuits de catégories différentes, concernant trois langues, et décrivons les informations qu'ils nous fournissent.

3.3.5.1 PILAF (Procédures Interactives Linguistiques Appliquées au Français)

PILAF [PILAF] est un analyseur morphologique et lemmatiseur gratuit du GETA, CLIPS. Il se trouve sur le web et on peut y accéder par une page web et donc par une requête http, ou en lui envoyant une requête dans un courrier électronique. Il est aussi possible de télécharger le logiciel (code source en C) et de le faire tourner sous Unix ou Windows.

PILAF prend une phrase française en entrée et produit en sortie les formes fléchies, les lemmes, les catégories grammaticales, et les variables grammaticales, sans désambiguïsation contextuelle syntaxique. Ainsi, dans l'exemple suivant, « une » donne deux résultats (article et article substantivé – « le » ne l'est pas : « une est venue » mais pas *« le est venu »).

Voici l'interface de PILAF et la sortie de la phrase « une cité retrouvera une zone côtière après un forum » :

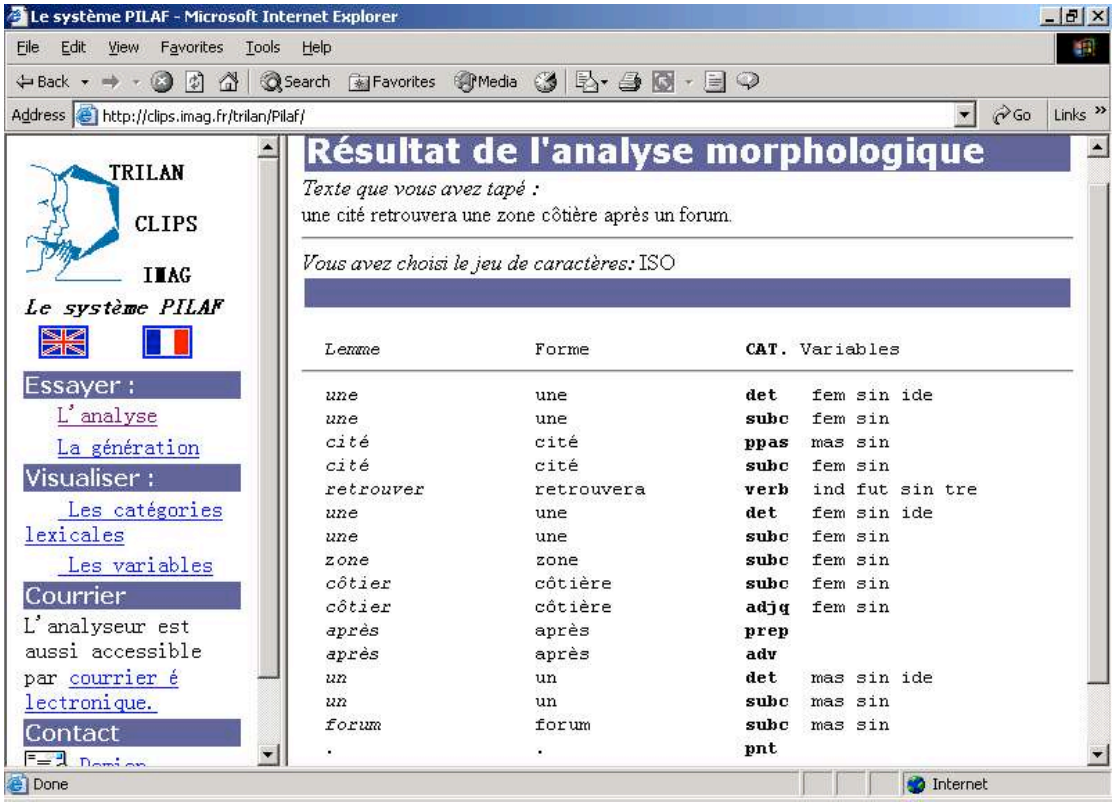


Fig. A-16 Interface du système PILAF

PILAF donne tous les lemmes candidats et leurs informations grammaticales, à savoir la catégorie grammaticale et les variables grammaticales. La liste complète de ces informations se trouve en Annexe D.

3.3.5.2 Autotag de CKIP

Autotag [Autotag CKIP _____] est un segmenteur et baliseur du chinois traditionnel développé par le groupe « Chinese Knowledge Information Processing » de l'Academia Sinica à Taiwan[3]. On peut le télécharger. Il y a deux versions, pour Unix et Windows. Il prend une phrase chinoise (simple ou complexe, terminée par un point chinois « _ ») en entrée, et la sortie est une phrase segmentée en mots avec les catégories grammaticales. L'analyse est basée sur un dictionnaire de 100.000 entrées intégré au programme.

Il est dommage qu'Autotag ne fournisse qu'un seul résultat de segmentation, parce qu'une phrase peut souvent avoir plusieurs segmentations. En plus, en chinois, la catégorie grammaticale n'est pas facile à juger : un mot peut être un verbe, un nom ou même un adjectif selon le contexte. Donc l'analyse de catégorie grammaticale a besoin d'une retouche plus précise. Par contre, le résultat de segmentation est assez correct.

Voici un texte chinois entré :

[3] L'auteur voudrait ici remercier le groupe CKIP pour l'avoir laissé utiliser ce logiciel

"_____"_____UNL_____
_____

Il est extrait du corpus UNL “UNLNews1”, obtenu par déconversion à partir d’un graphe UNL, et grammaticalement pas tout à fait correct.

Le texte original en anglais était :

The "Resolution in Suzhou" marks a turning point in the development of the UNL, both in terms of the strategic direction and in the management of its development and deployment.

Le texte traduit par la machine en français est :

La "résolution à Suzhou" marque un tournant dans le développement de l'UNL, en termes de direction stratégique et de gestion de son développement et de son déploiement.

Le résultat de segmentation est le suivant. Pour mieux comprendre, nous avons ajouté la prononciation et la signification de chaque mot chinois. Chaque mot chinois segmenté est donc suivi d’un triplet (catégorie grammaticale, prononciation, traduction française) :

1.__(PERIODCATEGORY)_"(FW)___(Na, jue2yi4, résolution)__(P, zai4, à)___(Nc, su1zhou1, Suzhou)_"(FW)___(VC, biao1ji4, marquer)__(Neu, yi1, un)____(Na, zhuan3zhe2dian3, tournant)__(P, zai4, à)_UNL(FW, unl, UNL)__(DE, de, de)___(VC, fa1zhan3, développement)__(COMMACATEGORY)

****

2.__(COMMACATEGORY)___(P, gen1ju4, selon)__(Nep, zhe4, ce)__(Nf, ge, classificateur)___(Na, zhan4lue4, stratégie)__(DE, de, de)___(Na, fang1xiang4, direction)__(Caa, he2, avec)__(P, zai4, à)__(Nh, ta1, il)__(DE, de, de)___(Na, fa1zhan3, développement)__(Caa, he2, avec)___(VC,bu4shu3, déploiement)__(DE, de, de)___(Na, guan3li3, gestion)__(PERIODCATEGORY)

****

(P pour préposition, Na pour nom commun, Nc pour nom géographique, VC pour verbe transitif d’action, Nh pour pronom, FW mot étranger, etc. Nous donnons une liste de ces catégories en Annexe D.)

Par rapport à PILAF, Autotag ne donne que les catégories grammaticales, puisqu’en chinois, il n’y a pas de conjugaison ni de déclinaison.

Voici l’interface d’Autotag. Le texte entré est dans le cadre du haut et le cadre du bas contient le résultat de segmentation :

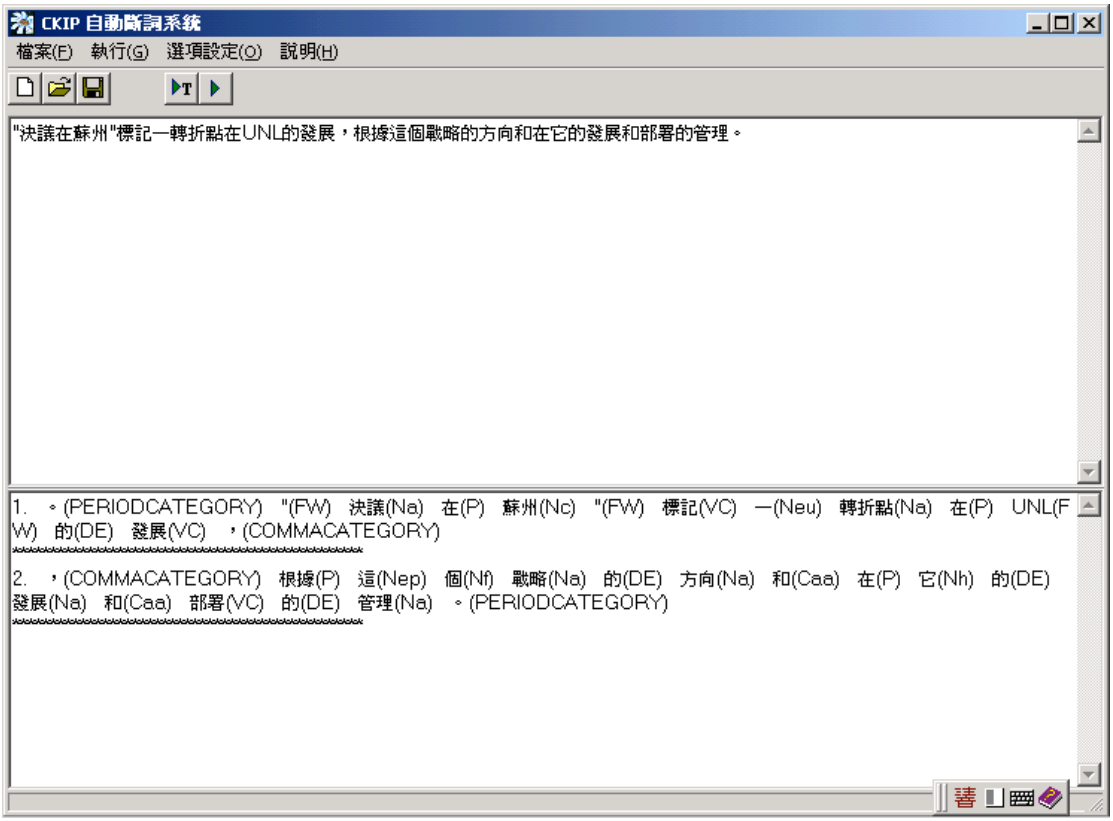


Fig. A-17 Interface du système Autotag

Les autres segmenteurs similaires du chinois sont Jasmine de l'université Chinoise de Hong Kong [Jasmine] et ICTCLAS (Institute of Computing Technology, Chinese Lexical Analysis System) [ICTCLAS]de l'Académie des Sciences Chinoise (CAS) à Pékin.

3.3.5.3 MeCab

MeCab est un analyseur morphologique du japonais développé par Université de Nara (NAIST). Il a été élaboré à partir de l'analyseur morphologique ChaSen et maintenant il est indépendant de ChaSen et a une vitesse plus élevée que ChaSen. MeCab s'exécute sur Unix et Windows. MeCab prend un énoncé en entrée. Sa sortie est textuelle et peut donc être enregistrée dans un fichier. L'utilisateur peut choisir une sortie sans ou avec les catégories grammaticales, et avec les N meilleures segmentations (N ≥1). Voici un exemple d'analyse sous Unix.

Le texte japonais entré est le suivant :

_____

La traduction en français est : « Taro a passé ce livre à la femme qui a vu Niro ».[4]

Nous donnons après chaque mot japonais segmenté par MeCab sa prononciation, sa catégorie grammaticale et sa traduction française : __ (tarou, nom propre, Taro) _

[4] Cette phrase se trouve sur le site web de MeCab, mais elle est probablement produite par la machine, car ce n'est pas une phrase normale en japonais. Le japonais correct pour exprimer la même signification est « _____ _ ».

(wa, postposition, marqueur d'agent) __ (kono, déterminant, ce) _ (hon, nom, livre) _ (wo, postposition, marqueur d'objet) __ (nirou, nom propre, Niro) _ (wo, postposition, marqueur d'objet) _ (mi, verbe, voir) _ (ta, auxiliaire, marqueur de l'action achevée) __ (jyosei, nom, femme) _ (ni, postposition, marqueur de cas datif) __ (watashi, verbe, passer) _ (ta, auxiliaire, marqueur de l'action achevée)

Voici un extrait de l'écran de MeCab sous Unix. La première ligne est la commande sous Unix ; la deuxième ligne est la phrase entrée et les suivantes donnent la sortie, avec sur chaque ligne un lemme et les informations grammaticales associées (catégorie grammaticale, sous-catégorie grammaticale, type de conjugaison, orthographe, orthographe en kana et prononciation).

```
% mecab
_____________________
__   __,_____,__,_,*,*,__,____,____
          *,*,*,*
_ __,____,*,*,*,*,_,_,_
          *,*,*,*,*
__   ____,*,*,*,*,*,__,__,__
_ __,__,*,*,*,*,_,{__/__},{__/__}
              *,*,*
_ __,____,__,*,*,*,_,_,_
                 *,*,*
__   __,_____,__,*,*,*,__,____,____
                 *,*,*
_ __,____,__,*,*,*,_,_,_
            *,*
_ __,__,*,*,__,____,__,_,_
         *,*,*
_ ____,*,*,*,_____,____,_,_,_
               *,*,*,*
__   __,__,*,*,*,*,__,_____,_____
               *,*,*
_ __,____,__,*,*,*,_,_,_
             *,*
__   __,__,*,*,_______,____,__,____,____
         *,*,*
_ ____,*,*,*,_____,____,_,_,_
_ __,__,*,*,*,*,_,_,_
EOS
```

Fig. A-18 Sortie de MeCab

Il y a d'autres analyseurs morphologiques gratuits du japonais sur le web. Ils offrent très souvent aussi les fonctionnalités de dictionnaires ou de romaniseurs.

Par exemple, ChaSen [ChaSen] de l'Université de Nara (NAIST), son prédécesseur JUMAN [JUMAN], de l'Université de Kyoto, et KAKASI [KAKASI] de Masahiko Sato à l'Université Hotoku.

3.3.5.4 Remarques sur les résultats d'analyse morpho-syntaxique

Il faut aussi remarquer que, à cause de la nature de langue et de la capacité des logiciels, les résultats donnés par ces segmenteurs/analyseurs morpho-syntaxiques sont des treillis.

Donc, pour utiliser correctement les résultats, il faut encore calculer le chemin le plus correct dans un treillis. Voici trois exemples en français, japonais et chinois.

Exemple (I) : « Je mange des pommes de terre. »

L'analyse de cette phrase nous donne le treillis suivant. Chaque nœud dans le treillis est un triplet « mot, catégorie grammaticale, lemme ».

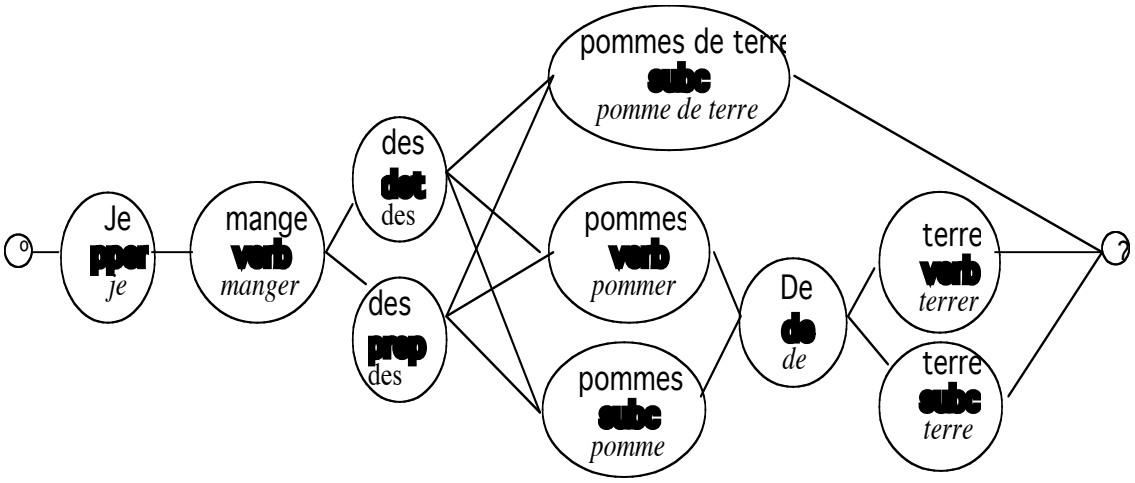


Fig. A-19 Analyse d'une phrase française en représentation par treillis

Exemple (II) : « _____ »

Selon la segmentation choisie, cette phrase japonaise peut avoir deux traductions :

« Veuillez enlever vos chaussures ici, s.v.p. » et

« C'est ici que vous êtes prié d'enlever votre kimono, s.v.p. ».

Dans la représentation en treillis, chaque nœud est un triplet (mot japonais, prononciation, traduction en français).

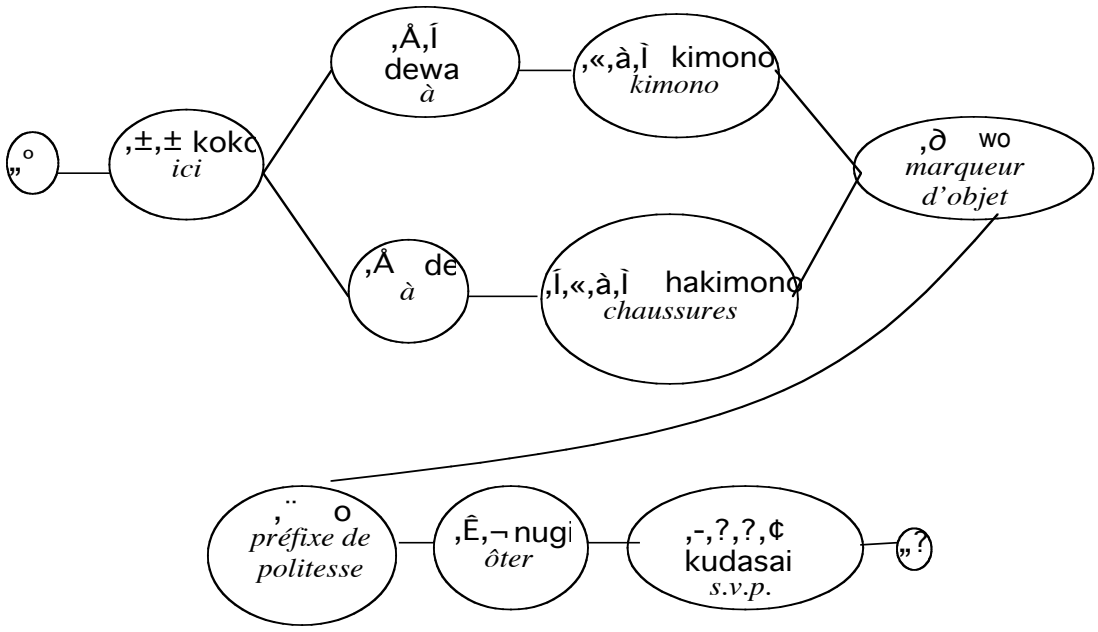


Fig. A-20 Sortie de MeCab en représentation par treillis

Exemple (III) : « _____ »

Cette phrase chinoise peut avoir au moins deux traductions selon le résultat de la segmentation : « Le parlement américain a donné son accord. » et « Les États-Unis

vont donner leur accord. ». Chaque nœud est un triplet (mot chinois, prononciation, traduction en français).

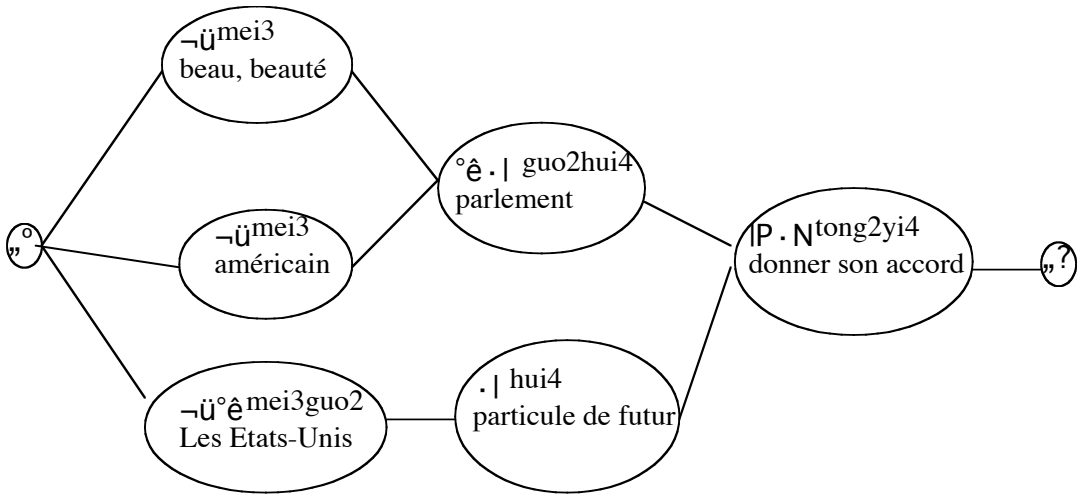


Fig. A-21 Analyse d'une phrase chinoise en représentation par treillis

La sortie de l'AMS est souvent un treillis, peu importe la langue analysée. Cela ne devrait pas être surprenant, étant donné l'ambiguïté de la langue naturelle.

Enfin, ce genre d'outil gratuit, utilisable via http ou téléchargeable, existe non seulement pour le français, le chinois et le japonais, mais aussi pour beaucoup d'autres langues, à commencer par l'anglais. Avec un peu de programmation de CGI (Commun Gateway Interface) ou des scripts en shell sous Unix, on peut facilement les intégrer dans un système de coédition. Par contre, on ne trouve que très peu d'analyseurs grammaticaux détaillés, et ce pour un très petit nombre de langues.

Et voici une liste non exhaustive :

TPD – baliseur de parties du discours
AM – analyseur morphologique
LM- lemmatiseur
SG – segmenteur
D – dictionnaire
usage : web (w)/ téléchargeable (t)

langue	nom	fonction	usage	url	commentaire
arabe, anglais, français, allemand, russe, etc.	Xerox	TPD/AM	w	http://www.xrce.xerox.com/competencies/content-analysis/toolhome.html	web page démos. licence à acheter
anglais, espagnol ,allemand, italien, langues nordiques	Connexor	TPD/LM/AM	w	http://www.connexor.com/demos/tqgger_en.html	version demo. licence à acheter
anglais, allemand,	TnT	TPD	w/t	http://www.coli.uni-sb.de/~thorsten/tnt/	Saarlandes University

espagnol					
anglais	Alembic	TPD	w	http://complingone.georgetown.edu/~sbj3/postagger.html	on peut entrer une URL (créé par MITRE)
anglais	ENGCG	TPD	w	http://www.lingsoft.fi/cgi-bin/engcg	Lingsoft de Finlande
anglais	Apple Pie	TPD/parseur	t	http://cs.nyu.edu/cs/projects/proteus/app	New York University
anglais	MXPOST	TPD	t	http://www.cis.upenn.edu/~adwait/jmx/jmx.tar.gz	(Statistics-Based)
anglais	LT POS	TPD	w	http://www.ltg.ed.ac.uk/software/posdemo.html	Edinburgh Language Technology Group (LTG)
anglais, allemand	QTag	TPD	t	http://web/bham.ac.uk/O.Mason/sftware/tagger/	1M mots entraînés pour anglais, 25K pour allemand (Birmingham University)
anglais, russe, suedios	Brill's	TPD	w	http://www.ling.gu.se/~lager/Home/brilltagger_ui.html	(Rule-Based Speech Tagger, source code du domaine public)
anglais	Brill	TPD	t	http://www.cs.jhu.edu/~brill/	Site web personnel d'Eric Brill (Microsoft Research)
anglais	OAK	TPD/parseur/AM	t	http://nlp.cs.nyu.edu/oak/	New York University
anglais	CLAWS	TPD	t/w	http://www.comp.ac.uk/ucrel/claws/	Lancaster Unicersity (UCREL)
allemand	Brill's	TPD/AM/LM	w	http://www.ifi.unizh.ch/CL/tagger/	Universität Zurich
allemand	Morphy	TPD/AM	t	http://www-psycho.uni-paderborn.de/lezius/morpho.html	Reinhard Rapp (FASK)
multilingue	MBT	TPD	w	http://www.ilk.kub.nl/~zavrel/tagtest.html	néerlandais,anglais,espagnol,suedois,allemand (Memory-Based Tagger)
français	FIPSTAG	TPD/parseur	w	http://www.latl.unige.ch	Université de Genève
français		TPD	w	http://www.atilf.fr	Insitut national de la langue (Nancy II)
latin		AM/D	w	http://www.perseus.tufts.edu/cgi-bin/morphindex?lang=Latin	Perseus Digital Library
grec		AM/D	w	http://www.perseus.tufts.edu/lexical.html	Perseus Digital Library
néerlandais	PoS	TPD	w	http://cosmion.net/jeroen/postag_index.html	by Jeroen Geertzen
portugais	Natura	TPD	w	http://natura.di.uminho.pt/natura/natura	
japonais	MeCab	TPD AM	t	http://cl.aist-nara.ac.jp/~taku-ku/software/mecab	NAIST
japonais	ChaSen	AM	t	http://chasen.aist-nara.ac.jp	NAIST
japonais	Juman	AM	w/t	http://www.kc.t.u-kyoto.ac.jp/ni-resource/juman.html	Kyoto University
chinois	Autotag	TPD/SG	t	http://godel.iis.sinica.edu.tw/CKIP/ws/	SINICA Taiwan
chinois/a	CEDI	D	t	http://www.cs.cmu.edu/~eep	encodage :BIG5 (25807

nglais	CT			eter/cedictb5.zip	mots le 30/05/2003)
chinois/anglais	CEDICT	D	t	http://www.cs.cmu.edu/~eepeter/cedictbg.zip	encodage :GB (25807 mots le 30/05/2003)
chinois ! anglais	VCDIC	D	t	http://ftp.iffcc.org/pub.software/ms-win/dict/vcdic350.exe	chinois/anglais, allemand
japonais/ multilingue	EDICT	D	t	http://www.csse.monash.edu.au/~jwb/j_edict.html	japonais/ anglais, allemand, français (XML et UTF8 encodage)
coréen	PSTech	TPD/AM	t	http://nlp.postech.ac.kr/~project/DownLoad/k_api.html	
thaï	Wsegol	SG	w	http://www.links.nectec.or.th/Wsegol	NECTEC
thaï/anglais	Lexitron	D	w	http://www.nectec.or.th/sll/R&D/te_mt.html	thaï _ anglais
indonésien/anglais	KEBI	D	w	http://nlp.aia.bppt.go.id/kebi/	Kamus Elektronik Bahasa Indonesia (BPP Teknologi)
multilingue	yourDictionary	D	w/t	http://www.yourdictionary.com	environ 800 dictionnaires en ligne
français/ multilingue	freelang	D	t	http://www.freelang.com/freelang/dictionnaire/index.html	français_langues étrangères 163 dictionnaires téléchargeables
multilingue	lexilogos	D	w/t	http://www.lexilogos.com/ressources.htm	dictionnaires électroniques pour télécharger
espérantо/anglais	Traduku	D	w	http://wwwtios.cs.utwente.nl/traduk/	espéranto_anglais/allemand/portugais/français/

Tableau A-3 Outils gratuits de traitement de langues naturelles sur Internet

B. Quel langage pivot choisir?

Introduction

Nous avons vu qu'un système de TA avec coédition était possible, à condition d'utiliser une forme « pivot » permettant la coédition de cette forme, i.e. son édition à travers le texte correspondant, dans une langue quelconque.

Il nous reste maintenant à définir le type le plus adapté de structure « pivot ». Nous présentons d'abord un état de l'art sur les pivots actuellement utilisés et utilisables en TA. Nous analysons aussi les différents types de pivots et les différents points de vue présentés dans des articles anciens mais pas obsolètes.

Nous conclurons en décidant d'utiliser UNL (Universal Networking Language) comme pivot, après avoir donné nos raisons et une introduction détaillée à UNL (projet, langage et format de document), son environnement, son état courant et les modules qui composent le système UNL.

Enfin nous développerons notre conception générale d'un système de coédition fondé sur UNL. Nous donnons plusieurs scénarios, puis les spécifications externes et internes d'une maquette de démonstration.

1. État de l'art sur les pivots utilisés et utilisables en TA

1.1 Introduction à la notion de pivot

1.1.1 Pivot architectural

Nous commençons par clarifier la définition de pivot et celle d'interlingua. Selon [Boitet 90d], un « langage pivot » est un ensemble de « structures intermédiaires » dans un système multilingue. Un pivot peut être une langue naturelle, plus ou moins contrainte, ou un langage abstrait totalement artificiel déconnecté des langues naturelles, ou n'importe quoi entre les deux. Le terme « pivot » a donc une valeur essentiellement « architecturale ».

L'architecture « pivot » en TA peut être représentée par la figure ci-dessous.

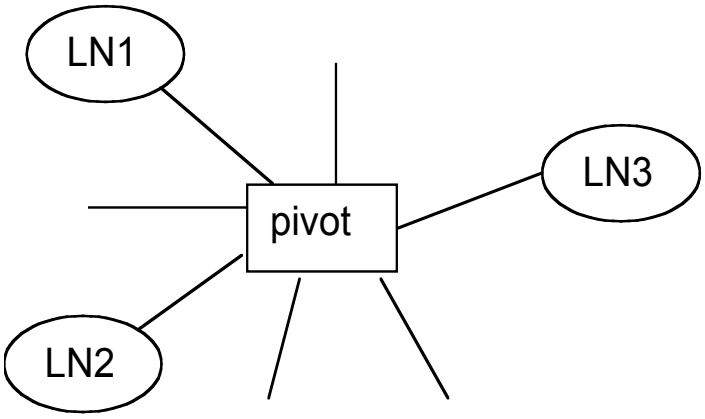


Fig. B-1 Architecture « pivot » d'un système de TA

Un énoncé d'une langue L_i est transformé en un énoncé « pivot » P, puis P est transformé en un énoncé dans les autres langues L_j.

1.1.2 Degré d'abstraction et de "sémanticité"

La différence entre un système de transfert et un système idéal à pivot peut être expliquée par la Fig. B-2 : un système de transfert comprend trois étapes principales : l'analyse, le transfert et la génération. Quand l'analyse n'est pas assez profonde, il faut passer par une étape de transfert. Plus profonde est l'analyse, moindre est le transfert. Dans un système idéal à pivot, la langue source est analysée jusqu'à la représentation intermédiaire indépendante, donc il n'y a pas d'étape transfert. Cela dit, une représentation indépendante est très difficile à construire et donc le pivot idéal reste toujours un peu théorique.

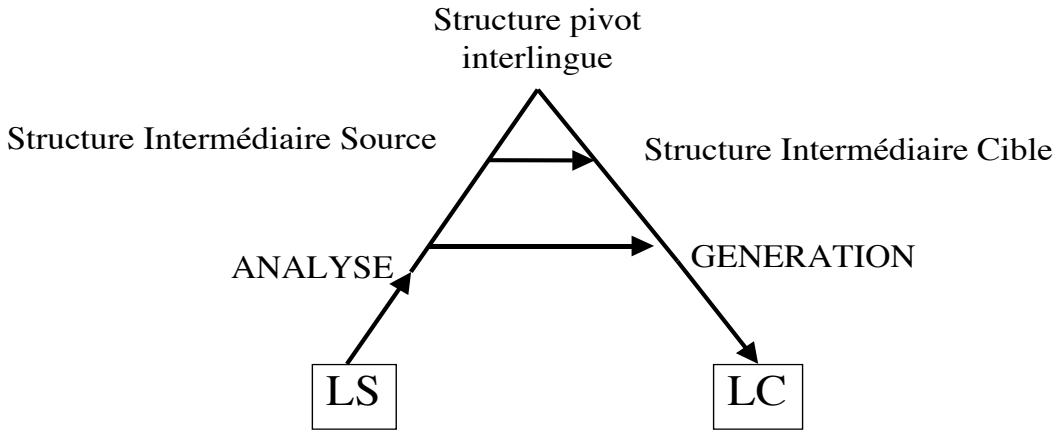


Fig. B-2 Système idéal à pivot

Plus précisément, la structure produite par l'analyse, P, peut dépendre de la langue L_i , par sa « façon de voir la monde », par exemple, par tel ou tel choix de relation sémantique, ou de précision lexicale. On peut alors être amené à effectuer ce que le Prof. Nagao a appelé un « transfert conceptuel » de P_i à P_j , avant de générer en langue L_j. Mais cette « adaptation » peut être laissée au soin du générateur de L_j, c'est-à-dire que P_j peut résulter de « préférences » s'appliquant à tout P_i , sans savoir à partir de quelle langue L_i il a été produit.

Un interlingua est un langage artificiel intermédiaire, qui est conçu pour exprimer tous les énoncés exprimables dans les langues naturelles traitées par le système, d'une manière neutre, indépendante de toutes les langues. Un interlingua doit avoir ses propres vocabulaire, relations grammaticales et attributs.

Ainsi, un interlingua est conçu pour être un pivot, mais un pivot n'est pas forcément un interlingua.

Nous trouvons dans la littérature au moins trois classifications de pivot et d'interlingua.

(I) Trois genres d'interlingua selon Tsujii [Tsujii 88]

- interlingua comme résultat d'une interprétation dans un domaine fixé, qu'on définit par ses concepts et son vocabulaire. C'est l'approche « top-down », ou « sémantico-pragmatique », utilisée par exemple dans le système KANT/CATALYST (CMU/Caterpillar) et dans les projets CSTAR et Nespole ! (« pivot IF »).
- interlingua comme une langue standard : on prend une langue naturelle et essaye de l'adapter (par exemple, par désambiguïsation) pour exprimer les énoncés dans les autres langues. C'est l'approche « bottom-up », utilisée par exemple dans le projet DLT (Distributed Language Translation, espéranto parenthésé).
- interlingua comme ensemble de primitives sémantiques : on définit un ensemble de primitives sémantiques et on les utilise pour exprimer les énoncés dans les autres langues, comme l'approche de « Conceptual Dependency » de Roger Shank.

Il manque ici une quatrième catégorie, celle des formalismes décrivant la structure abstraite (syntaxe profonde, niveau linguistico-sémantique) d'une langue donnée comme UNL (Universal Networking Language) [UNL].

(II) Trois genres de pivot selon Boitet [Boitet 86]

- Une LN : on utilise une langue naturelle ou même l'espéranto comme pivot avec ou sans des balises auxiliaires (parenthèses cachées).
- Un langage artificiel : on construit un langage artificiel comme pivot.
- Un pivot hybride à la Shaumyan : on définit les descriptions grammaticales universelles des langues mais on utilise le vocabulaire d'une langue naturelle.

(III) Deux genres d'interlingua selon Levin [Levin 02]

- Basé sur l'action de domaine
- Basé sur la sémantique lexicale

L'analyse de Levin n'est pas du tout complète, car il ne pensait dans cet article qu'aux représentations possibles pour des systèmes de TAO de dialogues finalisés.

Il existe aujourd'hui beaucoup de pivots et de systèmes à pivot, et des pivots de différents degrés d'abstraction et de sémanticité. Voici une liste de « pivots » possibles :

- Une LN « telle quelle ». Par exemple, on fait une « double traduction » via l'anglais pour faire du japonais-français.
- Une LN de grande diffusion « parenthésée » ou « balisée » (structure « concrète » plus ou moins riche).
- L'espéranto balisé comme dans le projet DLT, i.e. une langue artificielle proche de la langue naturelle mais moins ambiguë [Witkam 88] (variante du précédent, peu réaliste vu l'effort lexical, social et culturel nécessaire).
- Un pivot linguistico-sémantique comme UNL [UNL].
- une forme graphico-logique comme les graphes conceptuels de SOWA dans le système IBM [Conceptual Graphs (SOWA)].
- un pivot sémantico-pragmatique comme l'IF (Interchange Format) dans les projets CSTAR-II [CSTAR] et Nespole! [Nespole!].
- des formes logiques (logique classique, sémantique de Montague, Discourse Representation Theory, logique propositionnelle...) comme la grammaire de Montague dans le projet Rosetta [Odijk 89].

Pour notre recherche, nous ne faisons pas d'hypothèse a priori sur la sémanticité ou l'abstraction du « pivot ». De façon pragmatique, nous préférons étudier quelques systèmes à pivot et essayer de trouver le pivot qui nous convient.

1.2 Systèmes de TA utilisant l'architecture pivot et leurs pivots

Nous présentons maintenant plusieurs systèmes en détail pour avoir une compréhension plus profonde sur les interlingua et autres pivots utilisables en TA.

1.2.1 "PIVOT-I" du CETA (pivot "hybride" à la Shaumyan) (1963-1970) (propriétés et relations sémantiques et logiques)

Il s'agit de la toute première implémentation de l'architecture de TA à pivot.

1.2.1.1 Historique du système

Dans les années 1960, Y. Yngve a proposé un système de TA avec trois étapes logiques : une analyse monolingue, un transfert bilingue, et une génération monolingue. Le but de cette analyse monolingue est de produire pour chaque unité de traduction (une phrase à cette époque-là) une description structurale sans référence à la langue source. A partir de cette idée et en étudiant des théories avancées de linguistique (comme le modèle « sens - texte » de Mel'cuk et les « actants » de Tesnière), B. Vauquois a proposé « le langage Pivot-I » en 1963, un « pivot hybride » selon Shaumyan [Vauquois 69].

1.2.1.2 Description du pivot

Le PIVOT-I est une représentation logico-sémantique profonde qui représente un énoncé par un ensemble de prédicats munis de leurs argument et reliés entre eux par des méta-prédicats [Vauquois 74]. Ainsi, le résultat de la traduction ne dépend pas de la langue source. Ce pivot est « hybride », parce que d'un côté il a ses propres descriptions grammaticales universelles mais d'un autre côté il emploie le vocabulaire d'une langue naturelle (plus précisément les « unités lexicales » ou familles dérivationnelles) pour exprimer les concepts. Donc, dans l'application à la traduction automatique, il y a seulement un transfert lexical entre deux langues naturelles, réalisé en consultant un dictionnaire bilingue.

Il y a trois éléments pour le langage PIVOT-I : lexique (lié à chaque langue), variables et relations (interlingues).

Les unités lexicales appartiennent l'une des deux classes suivantes : éléments à valeur prédicative (verbes, substantifs verbaux, adjectifs, prépositions, conjonctions, etc.), ou éléments à valeur non prédicative (en général, les noms). En fait il s'agit ici de la classes de l'élément principal de l'UL. Par exemple :

> UTILE engendre la famille adjectivale (prédicative) UTILITE, (IN)UTILISABLE, (IN)UTILISABLITE, UTILEMENT□
>
> TERRE engendre TERREUX, TERRIEN, TERRESTRE□
>
> OBSERVER engendre OBSERVATEUR, OBSERVATION, OBSERVANCE, OBSERVABLE, OBSERVABILITE.

Quant à l'application à la TA, B. Vauquois appelait les variables interlingues du langage PIVOT-I « variables persistantes », car elles sont déduites de l'expression de la langue source et doivent être exprimées dans la langue cible pour conserver le sens. Ce sont, par exemple, la variable « énonciation » avec les valeurs « affirmative » et « négative » ; de même, le temps abstrait (TIME et non TENSE) et l'aspect, etc.

Les relations sont des métaprédicats du langage PIVOT-I, dont certains établissent la place des arguments des prédicats et les autres indiquent les relations entre les lexis ou leurs arguments. Toutes les relations utilisées sont des métaprédicats à deux places d'arguments.

Il y a 22 « métaprédicats » correspondant grosso modo à des « cas profonds » (notion introduite bien plus tard par Fillmore).

Un énoncé élémentaire est représenté par un prédicat (extrait du lexique) muni de ses arguments (extraits du lexique) et de variables portant sur le prédicat et les arguments. B. Vauquois appelait une telle représentation « une lexis ». Voici deux exemples de lexis :

« Le garçon porte un livre. »

Provenant de la lexis : Porter [PRED] (garçon [ACT1], livre [ACT2]).

« Le garçon est petit. »

Provenant de la lexis : Petit(garçon).

Pour construire la lexis dans le langage pivot, on dispose des relations qui placent les arguments dans une notion de prédicat :

Soit ACTn(a,P) où n=1, 2 ou 3

a est une unité du lexique

P est une unité du lexique à valeur prédicative.

Ainsi ACT1(a, P(x,y))=P(a,y)[5]

ou ACT3(c, P(x,y,z))=P(x,y,c)

On obtient la lexis Porter(garçon, livre) au moyen des deux relations

ACT1(garçon, Porter(x,y)) et ACT2(livre, Porter(x,y))

Parce que le PIVOT-I est basé sur la combinaison de lexis, il permet de reconnaître l'équivalence de phrases lorsque celles-ci diffèrent seulement par leur construction syntaxique, et de résoudre certaines ambiguïtés.

Ainsi, les phrases :

« Je possède cette maison »,

« Cette maison m'appartient »,

« Cette maison est à moi »

ne sont pas reconnues comme équivalentes par le langage PIVOT-I, parce que les 3 prédicats « posséder (x,y) », « appartenir_à (x, y) » et « être_à (x,y) » sont différents. On n'a pas dans le PIVOT-I d'équivalence du genre : (_x,y)[posséder(x,y)_ appartenir_à (y,x) _être_à (y,x)].

Par contre,

« Pierre écrit un livre »,

« Un livre est écrit par Pierre »

[5] l'actant 1* du prédicat P(x,y) est instancié par a. On le noterait en _–calcul (_x, P(x,y))(a).

sont considérées comme équivalentes, car elles donnent la même lexis : « écrire[Pred](Pierre[ACT1], Livre[ACT2]) ».

Bien sûr, par rapport à la technique actuelle, cette différence de style devrait être exprimée dans la langue cible.

1.2.1.3 Exemples du pivot

(I) « Le secrétaire n'a pas lu les journaux »

Cet énoncé élémentaire vient de la lexis LIRE(secrétaire, journal)

Avec « LIRE » porte des variables passé, perfectif, négatif

« secrétaire » porte des variables masculin, singulier, déterminé

« journal » porte des variables pluriel, déterminé

(II) « Le petit garçon porte un livre. »

Il y a deux lexis dans cette phrase : « L_1 : Le garçon est petit » et « L_2 : Le garçon porte un livre ». On utilise l'étiquette EPITHETE pour connecter garçon de L_2 et petit de L_1:

EPITHETE [Petit [ACT1 (garçon, Porter (x, livre))], ACT1(garçon, Porter(x, livre))]

1.2.1.4 Remarques

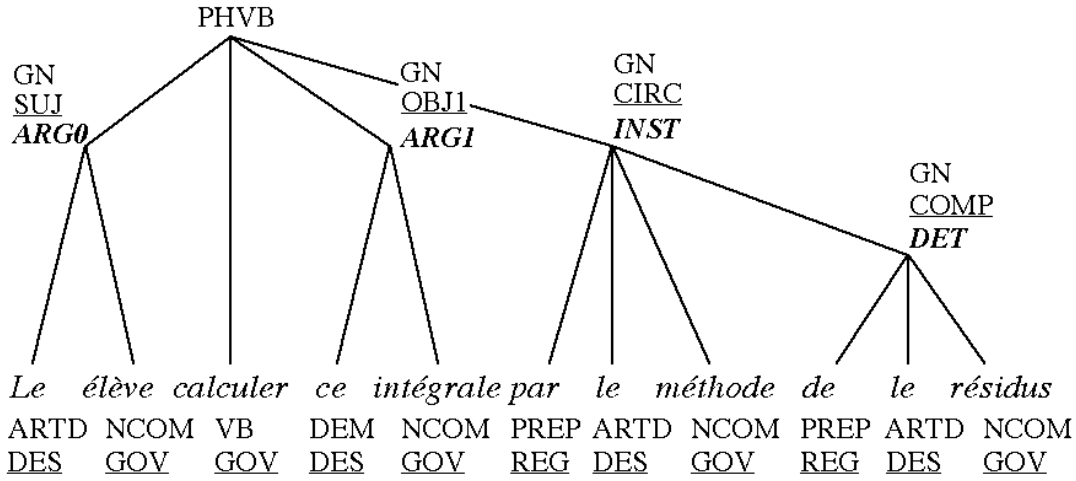
Le CETA a abandonné le PIVOT-I en 1970, sept ans après son invention. [Vauquois 85b] a expliqué pourquoi :

I. Il est très difficile de concevoir un tel pivot. Il n'y avait en effet pas une étude linguistique suffisante à cet époque-là. Il n'existait pas de description universelle pour le temps, la modalité, l'aspect, etc. En plus, le vocabulaire aurait dû être indépendant de toutes les langues naturelles, et malheureusement cela n'était pas le cas non plus.

II. Les traductions obtenues sont bien des paraphrases équivalentes, mais, sans indication relative à la surface, on ne peut pas obtenir le parallélisme de style demandé à une traduction professionnelle.

III. Problème du « tout ou rien » : si l'analyse ne produit pas un résultat complet et correct au niveau des relations sémantiques, ce qui est le point le plus difficile, on en est réduit à traduire mot à mot. En plus, si l'unité de traduction est plus grande qu'une phrase, il est presque sûr que le résultat d'analyse sera incomplet, et donc rien ne sera obtenu au niveau profond.

Dans la suite, pour augmenter la qualité maximale possible le CETA a adopté l'approche par « transfert multiniveau », reposant sur l'utilisation de m-structures (structures « multiniveau »).



légende :

CLASSE SYNTAXIQUE et SYNTAGMATIQUE,
FONCTION SYNTAXIQUE,
RELATION LOGIQUE et SEMANTIQUE.

Fig. B-3 Arbre d'analyse multiniveau

1.2.2 Titus IV de l'Institut Textile de France (1973-1995) (pivot fortement sémantique et LN contrôlée)

[Ducrot 82&88] Ce système repose sur un pivot associé à des langages contrôlés (un par langue) définis par une restriction très forte sur le vocabulaire et la syntaxe.

TITUS est cité dans plusieurs articles [Hutchins 95 & 99] [Boitet 82&88] comme un système à pivot fortement contrôlé, donnant un résultat assez satisfaisant, et fait sur mesure pour les résumés dans le domaine du textile.

1.2.2.1 Historique du système

[Boitet 76] [Zajac 88] Le système TITUS a été créé à l'Institut Textile de France en 1969. C'est un système documentaire multilingue (français, anglais, allemand, espagnol) pour l'indexage automatique et la traduction automatique des résumés stockés.

La traduction comporte deux phases : d'abord l'analyse de la langue source vers le pivot, et puis génération du pivot vers la langue cible.

Cette forme pivot est utilisée pour stocker les documents et rechercher l'information. La forme en LN est générée à chaque demande. Les résumés sont donc d'abord entrés par les documentalistes en langage contrôlé, puis stockés dans le systèmes en forme pivot.

Voici une figure du système TITUS-IV.

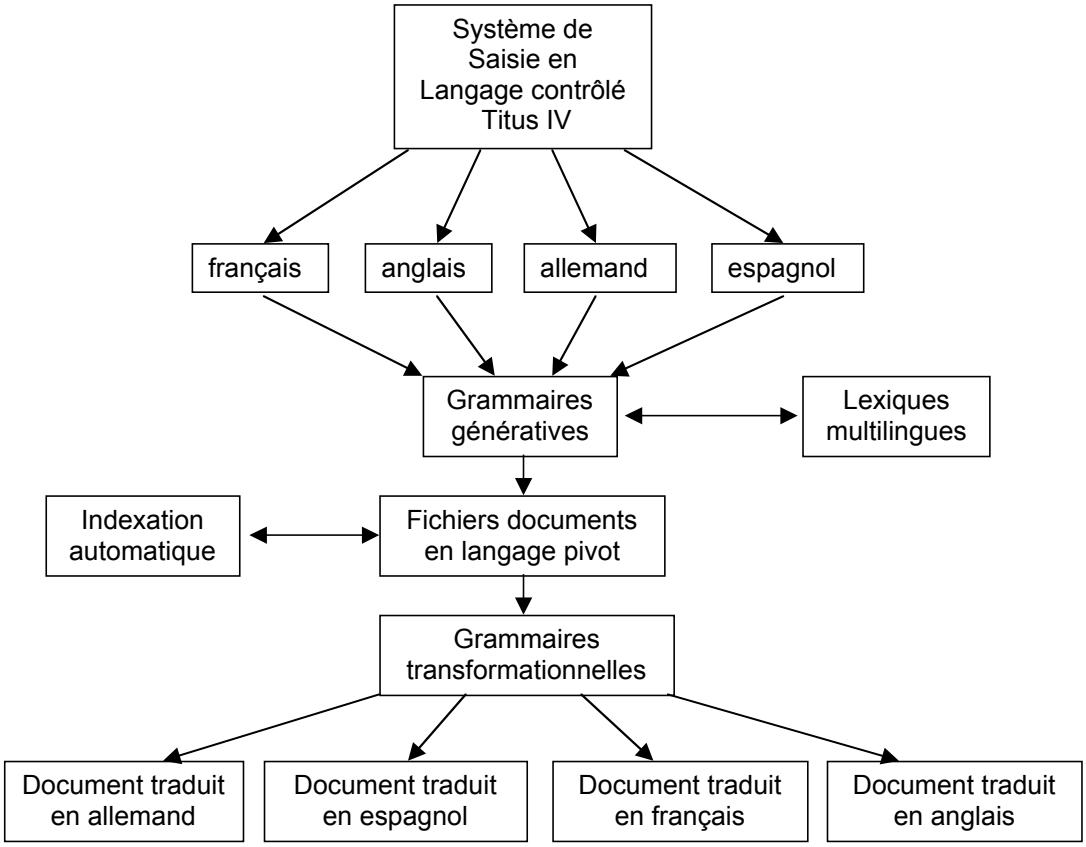


Fig. B-4 Structure de TITUS-IV

Dans le système TITUS, un résumé est entré au terminal de manière interactive en respectant les règles d'un langage contrôlé : un résumé est entré phrase par phrase, chacune devant être validée avant passer à la suivante. Le système TITUS-IV n'autorise qu'une proposition par phrase.

TITUS-IV est fondé sur une structure simple de la phrase, obéissant à des règles précises, valables pour toutes les langues indo-européennes et suffisamment souples pour permettre d'exprimer toutes les idées logiques courantes en langue naturelle.

TITUS-IV fut la dernière version de TITUS. TITUS-IV fut aussi adapté à des résumés en métallurgie (investissement de 4000 termes par le CNRS). Étant implémenté en assembleur 360 sur gros système IBM, il fut abandonné dans les années 80.

1.2.2.2 Description du pivot

[Zajac 88] Le vocabulaire de ce pivot se limite au domaine scientifique et technique car à l'origine le système était dédié à la traduction dans le domaine textile. Avec ce genre de vocabulaire très précis, on peut éviter au maximum l'ambiguïté lexicale des langues naturelles.

Le lexique de TITUS est multilingue. Comme un mot français peut correspondre à un ou plusieurs mots dans d'autres langues, la base du lexique de TITUS est l'Unité Lexicale (UL) et non pas le mot. Une entrée est composée de quatre parties contenant les unités lexicales équivalentes de chaque langue (un équivalent peut être constitué de plusieurs mots). Il y a quatre catégories d'UL : substantif, verbe, adjectif et

adverbe. Toutes les formes de conjugaison ou de déclinaison en chaque langue, et les informations grammaticales associées, sont enregistrées dans le lexique.

[Streiff 85] appelle le pivot de TITUS-IV « le swivel language » et dit que c'est un langage « binaire ». Cela veut simplement dire qu'il n'a qu'une forme interne, et pas de syntaxe externe le rendant lisible. Streiff n'a pas détaillé la structure de ce langage, il s'est borné à décrire le lexique contrôlé et la syntaxe des requêtes.

1.2.2.3 Remarque

C'est grâce à son domaine limité et à l'entrée fortement contrôlée qu'on obtient des résultats satisfaisants. Mais il y a deux défauts principaux [Zajac 88] : l'entrée en langage contrôlé n'est pas facile pour les utilisateurs ordinaires, de sorte que, dans la plupart des cas, ce sont les documentalistes qui entrent ou même réécrivent les résumés au lieu des auteurs, et il est possible que cela introduise des contresens.

Selon [Ducrot 88], le temps moyen consacré par un documentaliste pour rédiger un résumé de dix phrases (environ 100 unités lexicales ou 125 mots) est 10% plus élevé que pour l'écriture en langage naturel non contrôlé.

Deuxièmement, le langage pivot correspond à la finalité du système, mais il est aussi une limitation pour un système de traduction plus général. Selon [Streiff 85], le temps moyen pour apprendre la syntaxe des requêtes est d'environ 5 ou 6 jours pleins, ce qui est beaucoup trop pour le grand public.

Enfin, selon les documents et articles dont nous disposons, l'effort a surtout porté sur le contrôle de la syntaxe de l'entrée et sur la précision de la correspondance des lexiques entre les langues. Il n'est donc pas surprenant que son pivot n'ait pas été beaucoup décrit.

1.2.3 ALTAS-II de Fujitsu(1989-) (interlingua sémantique général)

1.2.3.1 Historique du système

Au début des année 1980, H. Uchida et K. Sugiyama, des laboratoires de Fujitsu, ont commencé à concevoir un système de TA japonais –>anglais basé sur une structure sémantique intermédiaire, appelée « structure conceptuelle » [Uchida 80].

Ils ont segmenté le texte japonais en « bunsetsu (__) » et puis utilisé la « grammaire des cas » de Fillmore pour marquer les relations entre les bunsetsus. Chaque phrase japonaise est représentée par un réseau sémantique.

Cette maquette fut d'abord testée sur un manuel d'utilisation d'un système informatique (environ 10 pages, 230 phrases au total). Le résultat fut satisfaisant.

Cette maquette fut le point de départ du très gros système ATLAS-II (Automatic Translation System). Son prédécesseur ATLAS-I était un système très différent, qui n'était pas un système à pivot mais un système direct destiné à la TA de japonais en anglais).

Cette maquette a été considérée très prometteuse pour la TA et la recherche d'information.

ATLAS-II était déjà assez modulaire en 1989 [Uchida 89]. Dans l'étape d'analyse, il y a un module SEGMENT qui s'occupe de l'analyse morphologique et un autre module ESPER qui s'occupe d'analyse syntaxique et sémantique. Dans l'étape de génération, il y a un mécanisme de fenêtre qui lit une partie de la structure conceptuelle et fait les opérations sur cette partie sous des conditions spécifiées.

Grâce à cette indépendance de la langue, ATLAS-II a été testé pour l'analyse et la génération du japonais, anglais, français, allemand, espagnol, chinois, swahili, et inuit (Eskimo) sans modification du logiciel de base. Mais la post-édition a toujours été indispensable et Uchida a estimé que la post-édition d'ATLAS-II prenait 30-50 % moins de temps que la traduction humaine (donc 45 à 30 minutes pour une page standard de 250 mots), donc qu'ATLAS-II était assez efficace.

Le système ATLAS-II a été commercialisé en 1982 pour les 2 couples EJ et JE. Sur une machine FACOM-M (système proche de Sun plus Unix), il pouvait traduire au maximum 60000 mots par heure, 240 fois plus vite que l'homme. Il était équipé d'un dictionnaire de base de 70000 termes dans les deux sens, et de 16 dictionnaires spécialisés pour un total de 250000 termes.

Fujitsu avait l'ambition d'inclure l'allemand, le coréen, et le français dans le système ATLAS-II commercial, mais finalement cela n'a pas été fait, bien que de gros prototypes aient été développés. Mais il n'y avait pas de marché suffisant. Aujourd'hui, le système ATLAS-II peut fonctionner sur un ordinateur personnel et le dictionnaire a été augmenté à plus d'un million d'entrées par langue.

En parallèle, de juillet 1983 jusqu'à février 1986, il y eut le projet coopératif SEMSYN-83 [Laubsch 84] [Rösner 86] entre Fujitsu et Siemens financé par le ministère de la recherche et de la technologie (BMFT) du gouvernement d'Allemagne de l'Ouest. Ce projet a utilisé l'analyse et l'interlingua d'ATLAS-II. L'équipe de Siemens a essayé plusieurs modules pour générer l'allemand à partir de la structure conceptuelle d'ATLAS-II, mais à la fin du projet, le système était toujours une maquette.

Plus tard, entre 1987 et 1993, le CICC (Centre of the Information Cooperation for Computerisation) [CICC] au Japon a mené un autre projet de système de traduction entre le japonais et quatre langues asiatiques (thaï, chinois, indonésien et malais). Le budget total a été d'environ 6G yen pour 7 ans.

Ce projet a permis le développement dans chaque langue d'un dictionnaire de base de 50000 termes et 25000 termes dans le domaine de l'informatique. Les règles de grammaire ont été conçues pour traduire un corpus de 3000 phrases. Le système repose sur la structure à pivot, qui emploie un interlingua adapté de celui de ATLAS-II, et il a utilisé la structure de dictionnaire et le vocabulaire conceptuel d'EDR (Electronic Dictionary Research) [EDR][Uchida & Chu 93] avec des symboles abstraits numérotés et des définitions en anglais.

1.2.3.2 Description du pivot

Au début du développement de ce pivot, Uchida a d'abord défini la structure conceptuelle qui est la structure intermédiaire sémantique entre deux langues naturelles. La structure conceptuelle est constituée de concepts et de relations. Un concept est représenté par un nœud et une relation est portée par un arc orienté reliant deux nœuds ou sortant d'un nœud (propriété). Un concept peut être général, spécial,

ou composé. Un concept composé est l'union de plusieurs concepts et relations, permettant d'exprimer un concept plus compliqué ou un concept qui n'existe que dans une langue.

L'unité de traduction est la phrase, et chaque phrase dans une langue naturelle correspond à une structure conceptuelle ou autrement dit à un réseau sémantique.

Uchida a défini quatre classes de concepts : verbe, adjectif, nom et adverbe. Il a aussi défini deux classes de relations : modificatrice de concepts et celle entre les concepts d'action et les autres concepts. En voici des exemples.

classe	Nom d'arc	explication
modificatrice	past, present, future	temps abstrait
	temporary, may, must	aspect ou modalité
Relations liant deux concepts	actor	qui fait
	object	objet d'une action
	property	propriété ou état d'une action ou d'un objet
	to	sens d'une action
	from	sens d'une action
	at	en même temps qu'une action
	after	avant une action
	before	après une action
	reason	raison d'une action

Tableau B-1 Relations semantiques du système ATLAS-II

Un arc peut être lié à deux concepts pour spécifier la relation entre les deux ou il peut être lié à un seul concept pour porter la modification sur ce concept.

Par exemple, la phrase « John drank » est exprimée par deux relations binaires : « drink -agent-> John » exprime la relation entre ces deux concepts « drink » et « John » ; «drink -past--> nil » exprime la modification sur le concept « drink ».

Au moment de l'analyse, ATLAS-II se réfère à un « modèle du monde (world model) » qui définit toutes les relations possibles entre concepts. Si le résultat d'analyse est conforme à ce modèle, le système accepte ce résultat, sinon le système demande une nouvelle analyse. Il n'y a pas d'interaction entre l'utilisateur et la machine pendant la procédure de TA.

Plus tard, dans le projet de CICC, ce pivot a été amélioré et les relations modificatrices ont été transformées en attributs. Un attribut sert à restreindre les concepts et à exprimer les perspectives et les intentions de l'énonciateur. Les relations entre propositions, comme conjonctive, subordonnée, coordonnée, etc. sont aussi prévues. Les pronoms ont été munis de la possibilité d'exprimer le genre, le nombre, l'exclusion ou l'inclusion d'interlocuteur, etc. Chaque concept est précédé par « #c ».

Dans les spécifications, on trouve 30 relations et 55 attributs. Les relations appartiennent à trois groupes : relations de cas, pseudo-relations, et relations sémantiques. Les attributs appartiennent à 6 groupes : les attributs qui restreignent la portée d'un concept, les attributs concernant l'aspect d'un événement, les attributs

temporels, les attributs concernant point vue de l'énonciateur, les attributs concernant ses intentions d'interlocuteur, et les attributs pour les éléments de phrase.

1.2.3.3 Exemples du pivot

voici un exemple de réseau conceptuel donné par [Laubsch 84]:

```
Structure conceptuelle :
((utterance – number -> one)
(purpose – number -> plural)
(want – obj -> achieve)
(want – pred -> *nil)
(*nil- st ->want)
(achieve – obj -> purpose)
(achieve – pred -> *nil)
(achieve – method -> utterance)
(achieve – agent -> speaker))

* le « st » à la 5ème ligne veut dire « le centre
sémantique (focus) de phrase »
```

```
Allemand: "Es wird gewuenscht, dass ein Sprecher mehrere
Zwecke mit einer einzelnen Aeusserung erreicht."
Anglais: "It is wanted that a speaker achieves several
purposes with one utterance."
```

1.2.3.4 Remarque

[Laubsch 84] indique que, dans le projet SEMSYN, le réseau sémantique venant du japonais est souvent sous-spécifié, et pas assez précis pour produire la phrase allemande.

Le système ATLAS-II est toujours disponible sur le marché et est le meilleur système de TA japonais _ anglais depuis 20 ans d'après les études de l'agence JEITA (ex JEIDA).

Son pivot est à l'origine du langage pivot UNL, utilisé dans le projet UNL, que nous verrons au 1.2.8.

1.2.4 PIVOT de NEC (1989-) (interlingua sémantique général)

[Miura 92] [Okumura 91] [Okumura 94]

1.2.4.1 Historique du système

Développé par NEC à partir de l'année 1983, PIVOT est un système de traduction japonais _ anglais. Son prédécesseur était le système VENUS, et PIVOT fut plus tard commercialisé sous le nom de « Honyaku Adaptor » (_____, adaptateur de traduction) puis de « Crossroad ».

Ce système a connu un assez grand succès commercial. Il utilise un pivot (sémantique) proche de celui d'ATLAS-II. L'interlingua comprend 49 relations

sémantiques. Dans le dictionnaire de base, il y a environ 40000 mots japonais et 53000 mots anglais ; pour le vocabulaire professionnel, il existe une vingtaine de domaines différents et dans chaque domaine il y a au moins 2000 mots. Pour l'analyse, il y a environ 3000 règles et 2500 pour la génération. La vitesse de traduction était de 6000 mots par heure, et le coût pour traduire le contenu d'une page de taille A4 (1400 signes ou 250 mots) était d'environ 1500 Yens (1994).

Au-dessus de la fonctionnalité de traduction, il existe aussi des modules pour aider les utilisateurs comme :

a. Traduction par batch et traduction de style oral.
b. Système de gestion et de manipulation de documents avec interface monolingue et bilingue.
c. Système de gestion et de manipulation de dictionnaires.
d. Gestion des mots inconnus et mise à jour des dictionnaires.

1.2.4.2 Aspect interactif dans le système PIVOT

[Miura 92] Le système fournit une interface interactive aux utilisateurs pour corriger les erreurs dans la représentation sémantique après que la phrase source a été analysée. Le système sauvegarde ces corrections comme données d'apprentissage pour que le système ne répète pas la même erreur.

Les erreurs que les utilisateurs peuvent corriger sont de 5 types :

a. dépendance
b. cas sémantique
c. phrases parallèles
d. portée
e. partage de concept

Voici un exemple de correction de dépendance :

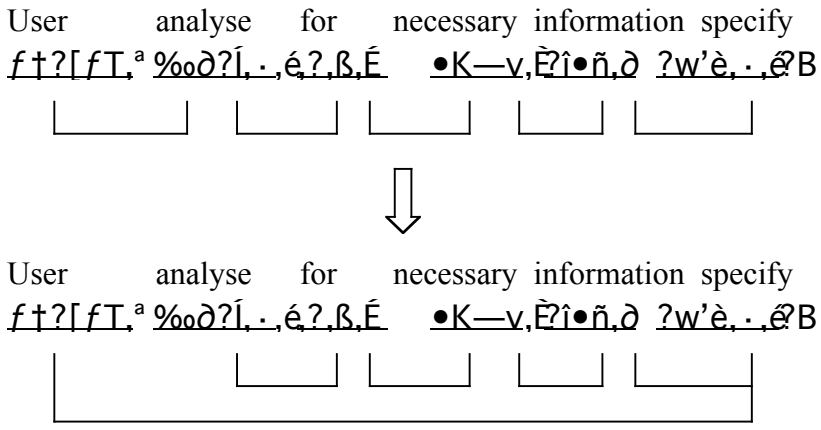


Fig. B-5 Correction de dépendance dans le système PIVOT

La traduction de la première phrase est : « On spécifie l'information nécessaire pour que l'utilisateur analyse ». La traduction de la deuxième phrase est « L'utilisateur spécifie l'information nécessaire à l'analyse ». Les deux analyses sont correctes, l'utilisateur est obligé de spécifier l'analyse de dépendance qu'il veut.

Voici un exemple de correction de cas sémantique :

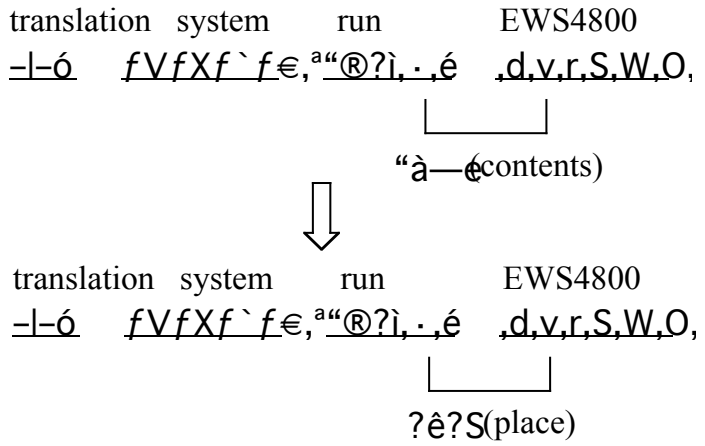


Fig. B-6 Correction de cas sémantique dans le système PIVOT

La première analyse est fausse, parce que le cas sémantique « __ » en japonais veut dire « apposition », et ici le substantif « EWS4800 » n'est pas équivalent à la phrase verbale (« le système de TA fonctionne »). Il faut donc changer le cas sémantique et la traduction de deuxième phrase est « EWS4800, où fonctionne le système de TA ».

Dans l'interface de cette interaction [Miura 92], l'utilisateur peut cliquer directement sur le texte pour corriger les erreurs. Pourtant, cette possibilité de correction n'est implémentée que pour les utilisateurs professionnels. En effet, il est difficile pour les utilisateurs de comprendre les cas sémantiques et l'analyse de dépendance.

Dans [Miura 92] l'auteur expose aussi des méthodes permettant de comparer la phrase avec les patrons stockés dans le système pendant l'analyse. Dans la suite, nous utilisons ce genre de méthodes pour créer des correspondances entre texte et structure pivot.

Comme PIVOT est un système commercial, et qu'il n'y a eu qu'une coopération universitaire avec la Thaïlande, on ne trouve en fait que très peu documents sur ce système.

1.2.5 Espéranto parenthésé/balisé dans le projet DLT (1982-1989) (LN+balises)

[Witkam 86] [Witkam 88] [Zajac 88] [Blanchon 94] [Schubert 88a]

1.2.5.1 Historique du système

Distributed Language Translation (DTL) fut un projet de traduction multilingue conduit par la société de services informatiques Buro voor Systeemontwikkeling (BSO) aux Pays-Bas. Le responsable en était Toon Witkam. Le système a été conçu en 1979 dans un environnement sans lien antérieur avec la TA. L'idée était d'utiliser

l'espéranto comme interlingua et de construire un système de TA multilingue. Après avoir déposé des brevets dans 14 pays, BSO a fait une étude de faisabilité de 1982 à 1983.

Le projet de 6 ans a commencé en 1984, il visait à produire un prototype d'au moins une paire de langues naturelles (anglais-français). Une démonstration a eu lieu en 1987 avec un petit vocabulaire de 2000 mots et une grammaire limitée. L'entrée était contrôlée, mais le but était de construire un système à entrée libre. Le projet a plus tard ajouté l'allemand et l'italien dans les langues visées. En 1988, l'idée de passer par un pivot fut abandonnée, alors que les résultats étaient prometteurs, et le groupe se tourna, sans succès, vers des méthodes utilisant des mémoires de traduction (bitexte). Le projet s'est terminé en 1992.

L'analyse génère d'abord tous les arbres de dépendance possibles de la langue source. Puis, en remplaçant le vocabulaire en espéranto et en appliquant les règles de metataxis (pour transformer certaines structures syntaxiques de langue source en celle de l'espéranto), le système obtient les représentations en espéranto. Pour évaluer ces arbres, le système emploie le module SWESIL (Semantic Word Expert System for the Intermediate language) qui, en consultant une LKB (Lexical Knowledge Base), calcule un score pour chaque représentation. C'est une idée très proche de celle d'ATLAS-II.

Cette LKB stocke les paires de mots les plus fréquentes (avec les relations de dépendance) et sert de base au calcul sémantique. Pour tous les nœuds qui ont des relations dépendantes, on vérifie si ces relations sont enregistrées dans la LKB. A la fin du projet, la LKB comprenait environ 75000 paires de mots.

Par exemple, le mot « couper » a deux candidats possible en espéranto : « tondi » et « tranchi ». Tranchi veut dire en peu près trancher et tondi tondre, découper. Si on veut traduire « couper le gâteau » et si malheureusement la paire « tranchi-kuko » n'existe pas dans la LKB, mais on y trouve les paires « tranchi-pano (couper-pain) » , « tondi-papero (couper-papier) », et « tondi-herbo (couper-herbe) », le système parcourra ces trois paires et calculera la distance sémantique entre gâteau-pain, gâteau-papier et gâteau-herbe. Comme gâteau est plus proche de pain, c'est le verbe « tranchi » qui sera choisi.

Si nécessaire, DLT utilise ensuite une phase de désambiguïsation interactive avec l'utilisateur. A la fin une seule représentation IL (« interlingvo », interlingua en espéranto) est choisie.

En suite, l'arbre IL est transformé en texte IL, et mis sur le réseau ou envoyé à une station réceptrice pour produire la langue cible plus tard. Le texte IL est un texte espéranto balisé (mais les balises sont cachées au lecteur normal).

Un aspect essentiel et novateur de DLT est que son architecture est distribuée. Il y a des ordinateurs connectés sur un réseau, et la station qui s'occupe de l'encodage n'est pas forcément la même que la station réceptrice.

1.2.5.2 Description du pivot

Dans le projet DLT, l'espéranto légèrement modifié est utilisé comme interlingua. La raison pour laquelle l'espéranto a été choisi pour l'interlingua était double, politique et scientifique. La première était à vrai dire meilleure que la seconde : il s'agissait de

promouvoir l'utilisation de l'espéranto, et de créer d'importantes ressources pour lui (dictionnaires, analyseur, générateur, traducteurs, LKB, base de corpus « bitextes » espéranto-LN_x). La seconde est beaucoup plus douteuse, et en fait pseudo-scientifique et erronée : on prétendait que l'espéranto était rigoureux et non ambigu, mais toute langue naturelle, même construite consciemment, sécrète l'ambiguïté par son seul usage.

Selon le principe de DLT, la désambiguïsation ne se fait que par des moyens linguistiques, sans ajouter des numéros, index, parenthèses, étiquettes, etc. La seule exception est l'insertion d'un espace de plus pour indiquer que le dépendant qui suit l'espace ne dépend pas du mot qui le précède, mais du mot qui précède son prédécesseur. On verra une tel exemple ci-dessous. [Schubert 86]

La modification apportée à l'espéranto est destinée à réduire l'ambiguïté syntaxique. Selon [Witkam 88], cette modification n'était pas si grande qu'elle avait été prévue et l'espéranto modifié est encore facile à lire. Il s'agissait d'ajouter des étiquettes (par exemple : _, ‘, espace) dans les mots et les phrases en contrôlant certains aspects de la syntaxe pour désambiguïsation et de définir plus précisément les usages de certains mots et les règles grammaticales ambiguës. En espéranto, cet interlingua est appelé “interlingvo (IL)”. Nous donnons plus loin quelques exemples pour voir la différence entre l'espéranto et cet IL ; la description détaillée sur cet IL se trouve dans [Interlingvo] (en espéranto).

Les modifications sont de 4 classes :

-*(I) Morphologique* – pour préciser la limite d'un morphème, pour définir plus clairement les nouvelle morphème, paradigme de conjugaison, de déclinaison, et les mots grammaticaux.

Lexicalement, on insère dans les mots des apostrophes pour distinguer les limites des affixes, par exemple (« E-o » abrège ici « Esperanto ») :

> Français : avenir
> E-o : estonteco
> IL : est'ont'ec'o (« est » racine pour « être », « ont » participe actif futur[6],
> « ec » affixe de substantivation, « o » affixe de nominatif)

En espéranto, il y a deux possibilités de former la voix passive, soit par le verbe « estas » (« être » en français) plus participe passif ou le verbe à la forme passive. Le verbe à la forme passive est formé par « participe passif + suffixe de verbe ». Les espérantistes en fait disputent encore la légitimité de ce genre de verbe. Dans IL, ce genre de verbe est légal, il est caractérisé par l'affixe inventé « ajt ».

> IL: man_'ajt'int'as (racine « manger »+affixe du verbe passif+participe actif passé+affixe du temps présent)
> E-o:man_itas (=estas man_ita)
> Français : avoir eu été mangé

[6] En espéranto on distingue 6 participes : il y a deux voix (actif et passif) et trois temps (présent, passé et futur).

IL:man_'ajt'ont'is (racine « manger »+affixe du verbe passif+participe actif future+affixe du temps passé)
E-o:man_otis (=estis man_ota)
Français : allait être mangé

-*(II) Vocabulaire* – pour définir les nouveaux mots déjà utilisés en espéranto, la création de nouveaux mots, en ajoutant quelques nouveaux affixes et en limitant l'utilisation de certains affixes.

Français : au lieu de (préposition / conjonction de subordination)
IL : anstata_ (préposition) / anstata__ke (conjonction de subordination)
E-o : anstata_ (préposition / conjonction de subordination)

Français : à, en, etc.
IL : iam_en (préposition temporelle) / ie_en (préposition locative)
E-o : en (préposition temporelle et locative)

Français: Il vole vers Hambourg.
IL: _i flugas ie_al Hamburgo
E-o: _i flugas Hamburgon

(En espéranto les cas datif et accusatif sont tous les deux marqués par « –n », ce qui montre qu'il y a des ambiguïtés même au niveau des flexions, contrairement aux affirmations initiales du projet. Dans cet exemple, pour distinguer que Hambourg est la destination, une nouvelle étiquette « ie_al » est utilisée.)

- *(III) Syntaxique* – pour bien séparer des mots qui appartiennent en même temps à plus d'une catégorie, spécifier les règles de l'accord, de l'ordre des mots, et de l'ellipse.

Français : Je le voyais sur le bateau.
E-o : Mi vidis lin sur la _ipo.
IL : Mi vidis lin _sur la _ipo.

Dans cet exemple, le français et l'espéranto sont ambigus. En IL il faut insérer un espace devant « sur » pour indiquer que ce mot n'est pas dépendant du mot devant lui. Et donc « sur » est dépendant de « vidis (voyais) », cela indique que c'était moi qui étais sur le bateau quand je le voyais.

-*(IV) Sémantique* – Définir la portée des verbes modaux, par exemple:

« Ili nun devas jam esti ie-en Romo (ils/ maintenant/ doivent/ déjà/ être/ à/ Rome) » ne peut pas dire en IL « je suppose fortement qu'ils sont maintenant à Rome », mais plutôt seulement « Ils devraient être maintenant à Rome (mais à cause du retard du train ils sont pas encore arrivés) ».

Pour dire « Je suppose fortement qu'ils sont maintenant à Rome » en IL, on dira : « Deve, ili nun jam estas ie-en Romo (Sans doute/ ils/ maintenant/ déjà/ être/ à/ Rome) » (en utilisant l'adverbe au lieu du verbe modal pour exprimer la certitude ou le souhait concernant la phrase entière).

1.2.5.3 Exemples du pivot

Nous trouvons dans [Schubert 86], l'exemple suivant :

> Anglais : « Many multinationals were allocated grants for the study of capital development strategies for Third World member states, which will be of increasing importance in the future. »
>
> IL : « Al mult'a'j mult'naci'a'j entrepren'o'j asign'ajt'is subvenci'o'j por stud'o de strategi'o'j'n por per'kapital'a evolu'ig'o _por tri'a'mond'a'j _tat'o'j-membr'o'j_, kiu'j hav'os kresk'ant'a'n grav'ec'o'n iam-en la est'ont'ec'o.

*1. En espéranto comme dans beaucoup d'autres langues, l'objet indirect ne peut pas être le sujet d'une phrase passive. Donc ici « Many multinational » ne peut pas être le sujet. Il faut d'abord effectuer le transfert de l'arbre anglais en appliquant une règle de métataxis. Le résultat est que la préposition « al » est ajoutée devant le sujet et que la phrase devient passive, avec le verbe à la voix passive.

*2. Tous les mots en IL sont apostrophés pour bien distinguer les limites des affixes.

*3. Remarquons que la préposition dans « iam-en la est'ont'ec'o » (à l'avenir) est « iam-en », c'est l'emploi temporel d'« en ».

*4. La marque de soulignement après « memnbr'o'j » signifie que le mot « kiu » qui le suit dépend de « _tat'o'j » mais pas de « memnbr'o'j ». (Dans le IL, cette marque de soulignement est simplement une espace de plus, nous utilisons ici une marque de soulignement pour l'exprimer explicitement).

1.2.5.4 Remarque

Après plus de cent ans d'utilisation, l'expressivité de l'espéranto est sans doute la même que celle d'une langue naturelle, avec plus de régularité et de simplicité. Pourtant, comme toutes les langues naturelles, l'espéranto n'est pas un bon candidat pour un interlingua. Même les structures profondes de l'espéranto ne conviendraient pas pour deux raisons :

> trop grande complexité des structures de syntaxe profondes de toute langue.
>
> extrêmement faible diffusion de l'espéranto, et donc incompréhension de son lexique par les développeurs.

Une autre tentative d'utiliser une langue naturelle comme interlingua a été faite dans le système ATAMIRI [Guzman de Rojas 88]. Le projet a utilisé une langue indienne parlée en Bolivie, l'aymara. Selon l'auteur, cette langue utilise une logique à trois valeurs. Le système a été prototypé sur le couple anglais-espagnol.

Une autre tentative plus radicale serait d'utiliser un langage totalement inventé, c'est ce que propose Lobjan [Nicolas 96]. Mais cela n'a pas encore été testé dans un projet expérimental.

1.2.6 KBMT-89 (par CMU) (1987-1989) (Interlingua général avec ontologie)

[Goodman 89] [Goodman 92] [Blanchon 94] [Nirenburg 90] [Nyberg 92] [Nyberg 97] [Lonsdale 94]

La différence principale du système KBMT-89 est qu'il repose fortement sur l'exploitation d'une « ontologie » (autrement dit sur un modèle du domaine lié à un lexique conceptuel [Nirenburg 89]) dans la procédure de traduction.

1.2.6.1 Historique du système

Le projet KBMT-89, développé par le centre de TA à CMU (Carnegie Mellon University), avait pour but de construire une maquette de TA avec le paradigme de pivot dans le domaine de la traduction et la maintenance de manuel d'ordinateur personnel (le PC anglais et le PC-550 « japonisé » par IBM), en employant un modèle du domaine. Les langues sources et cibles étaient l'anglais et le japonais.

La taille du vocabulaire était plutôt petite : pour l'analyse, 800 termes de japonais et 900 termes d'anglais ; pour la génération, 800 termes de japonais et 900 termes d'anglais. Il y a environs 1500 concepts dans l'ontologie. Le formalisme de la grammaire est basé sur la Grammaire Lexico-fonctionnelle (LFG, Lexical-Functional Grammar). La représentation syntaxique est en structure fonctionnelle (f-structure). Toutes les représentations de connaissance sont exprimées à l'aide du système FRAMEKIT, y compris les concepts, le vocabulaire, les règles de transfert vers/de l'ILT (MR, Mapping Rules), et le langage pivot (ILT, Interlingua Text)

Voici une figure qui montre l'architecture du système KBMT-89 [Goodman 89] [Blanchon 94]

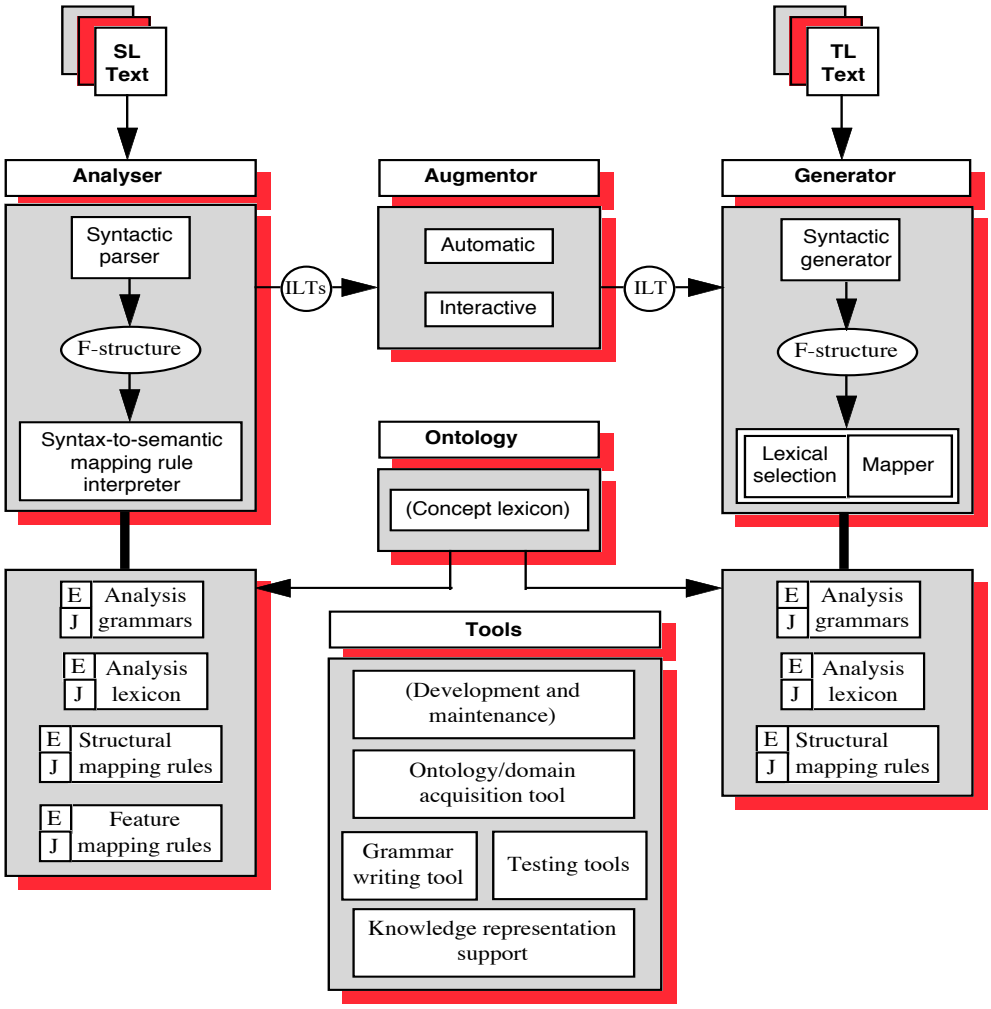


Fig. B-7 Structure du système KBMT-89

Le système inclut les composants suivants :

un analyseur syntaxique avec un interpréteur de contraintes sémantiques ;

un module d'application de contraintes sémantiques ;

un désambiguïseur interactif : l'Augmentor ;

un générateur sémantique produisant la structure syntaxique dans la langue cible et effectuant la sélection lexicale ;

un générateur syntaxique produisant le texte dans la langue cible ;

un modèle du domaine (ontologie) ;

des outils pour le développement et la maintenance des concepts, de la grammaire, et du vocabulaire.

La désambiguïsation interactive est mise en œuvre quand l'ILT est ambiguë ; le système pose des questions de clarification dans la langue de l'interface. Les détails de l'Augmentor sont expliqués dans [Blanchon 94].

Voici une figure qui montre l'interaction entre l'utilisateur et le système :

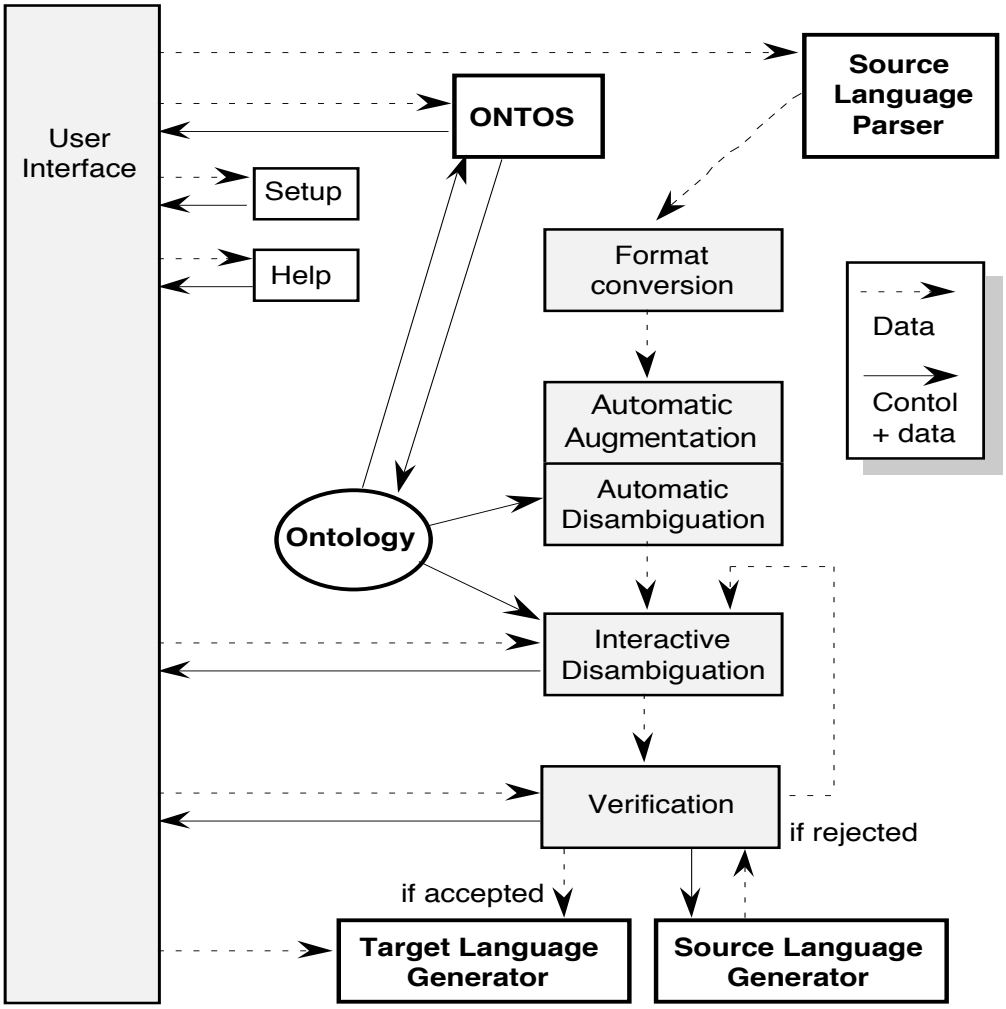


Fig. B-8 Interaction entre utilisateur et système KBMT-89

Les interactions entre l'utilisateur et le système incluent les points suivants :

L'utilisateur fournit le texte en langue source au parseur ;

L'utilisateur développe et maintient l'ontologie par l'outil ONTOS ;

L'utilisateur participe à la désambiguïsation interactive ;

L'utilisateur vérifie le résultat de la désambiguïsation ;

L'utilisateur donne des commandes aux modules de SetUp et Help.

La version finale de KBMT-89 a été démontrée en février 1989 [Blanchon 94].

La structure et la conception de KBMT-89 ont été ensuite exploitées dans les projets KANT, Pangloss, et Mikrokosmos. En 1991, CMU a commencé le projet KANT (Knowledge-based Accurate Natural-language Translation), qui était le prolongement de KBMT-89 [Nyberg 97], pour la société CATERPILLAR .

L'exploitation a commencé en 1994 pour produire les documentations multilingues des équipements lourds de CATERPILLAR [Mitamura 93], et KANT a été renommé CATALYST.

KANT prend l'anglais contrôlé (Constrained Technical English) [Lonsdale 94] comme langue source et produit la sortie en français, espagnol, allemand, italien, portugais et chinois [Nyberg 97]. A part l'exploitation d'anglais contrôlé, le SGML (Standard Generalized Markup Language) est aussi utilisé en entrée. Des balises pour spécifier la structure sémantique et logique sont définies dans une DTD (Document Type Definition) pour la désambiguïsation. Les informations dans ces balises sont ensuite plus tard analysées par le parseur et ajoutées à l'interlingua.

KANT est maintenant un système qui comprend une série d'outils, de logiciels et une couverture de 65000 concepts (dont 2000 concepts d'action) et 35 structures d'argument. Le résultat est assez satisfaisant, avec une postédition minimale [Nyberg 97]. Cependant, sur les 11 langues cibles prévues, 4 seulement sont opérationnelles.

1.2.6.2 Description du pivot

Le système FRAMEKIT sert à toutes les représentations de connaissance de KBMT-89, elles sont exprimées par des structures de cadres (frames). Un cadre peut avoir une ou plusieurs cases (slot) ; une case peut avoir une ou plusieurs facettes (facet) ; une facette peut avoir une ou plusieurs vues et un ou plusieurs remplisseurs (filler).

Voici un exemple de structure de cadre :

```
;KBMT-89 report p.25
*(make-frame
      cmu
            (is-a (value(common university non-profit-institution)))
            (location   (city (common Pittsburgh))
                        (state (common pa))
                        (country (common usa))))
```

Le nom du cadre est « cmu » ; il y a deux cases : is-a et location ; la case « location » a trois facettes et « is-a » en a une. Toutes les vues sont communes, ce qui veut dire que ces vues sont visibles par tout le monde. Enfin, la facette « value » a deux remplisseurs (« university » et « non-profit-institution ») et les autres en ont un.

Le dictionnaire de KBMT-89 est composé de cadres. Chaque cadre représente une entrée et spécifie les informations de cette entrée. Il lie aussi cette entrée avec un concept dans l'ontologie.

Voici quelques exemples d'entrées dans les dictionnaires anglais et japonais : dans ces exemples, nous voyons que le verbe anglais « remove » et le verbe japonais « torinozoku » sont liés au concept « remove », tandis que le nom anglais « tape » et le nom japonais « teepu » sont liés au concept « sticky-tape ».

```
;Un exemple de verbe anglais
;KBMT-89 report p.98
("remove" (CAT V)
   (CONJ-FORM INFINITIVE)
   (FEATURES
      (CLASS DEFAUT-VERB-FEAT)
      (all-features (*OR*
            ((FORM INF)(VALENCY TRANS)(COMP-TYPE NO)
             (ROOT REMOVE))
            ((PERSON (*OR* 1 2 3))(NUMBER PLURAL)(TENSE PRESENT)
             (FORM FINITE)(VALENCY TRANS)(COMP-TYPE NO)
```

```
             (ROOT REMOVE))
            ((PERSON (*OR* 1 2))(NUMBER SINGULAR)(TENSE PRESENT)
             (FORM FINITE)(VALENCY TRANS)(COMP-TYPE NO)
             (ROOT REMOVE)))))
   (MAPPING
     (local
          (HEAD  (REMOVE)))
     (local
          (slots (SOURCE=(PPADJUNCT (PREP=FROM)))))
     (CLASS CB-TH-VERB-MAP)))


;un exemple de nom anglais
("tape" (CAT N)
   (CONJ-FORM SINGULAR)
   (FEATURES
     (CLASS DEFAULT-NOUN-FEAT)
     (all-features ((PERSON 3)(NUMBER SINGULAR)(COUNT YES)
(PROPER NO)(MEAS-UNIT NO)(ROOT TAPE))))
   (MAPPING
     (local
          (HEAD (STICKY-TAPE)))
     (CLASS OBJECT-MAP)))


;un exemple de verbe japonais
("torinozoku" (CAT V)
   (MAPPING
     (local
          (HEAD (REMOVE)))
     (CLASS AGENT-THEME-MAP)))


;un exemple de nom japonais
("teepu" (CAT N)
   (MAPPING
     (local
          (HEAD (STICKY-TAPE)))
     (CLASS OBJECT-MAP)))
```

Le pivot (ILT, Interlingual Text) de KBMT-89 est aussi composé de cadres, qui représentent le résultat d'analyse de la grammaire LFG d'un énoncé de l'anglais ou du japonais. Les relations entre les énoncés ne sont pas exprimées. Un énoncé en langue naturelle est exprimé par plusieurs cadres selon le nombre de rôles sémantiques et de propositions dans cet énoncé. Le centre sémantique est stocké dans le cadre « proposition » et chaque cadre de rôle sémantique est attaché à ce cadre. Il y a un cadre « acte de parole (speech-act) » pour exprimer les autres informations concernant cette proposition, et enfin il y a un cadre « clause » au-dessus des cadres « acte de parole » et « proposition ».

Voici une figure qui explique la procédure de transformation de texte dans le système KBMT-89 ([Goodman 89] *KBMT-89 report p.7*) : la langue source est dans ce cas le japonais. Le texte japonais est d'abord analysé par un parseur et transformé en ILT (interlingual text). Ici ILT est une représentation schématique qui montre qu'il y a 6 cadres (clause, préposition, acte de parole et 3 rôles sémantiques) dans cet ILT. Dans le cadre de clause ; on stocke toutes les informations de cette clause et le nombre de

propositions dans cette clause, chaque proposition a un cadre d'acte de parole et un ou plusieurs cadres de rôles sémantiques.

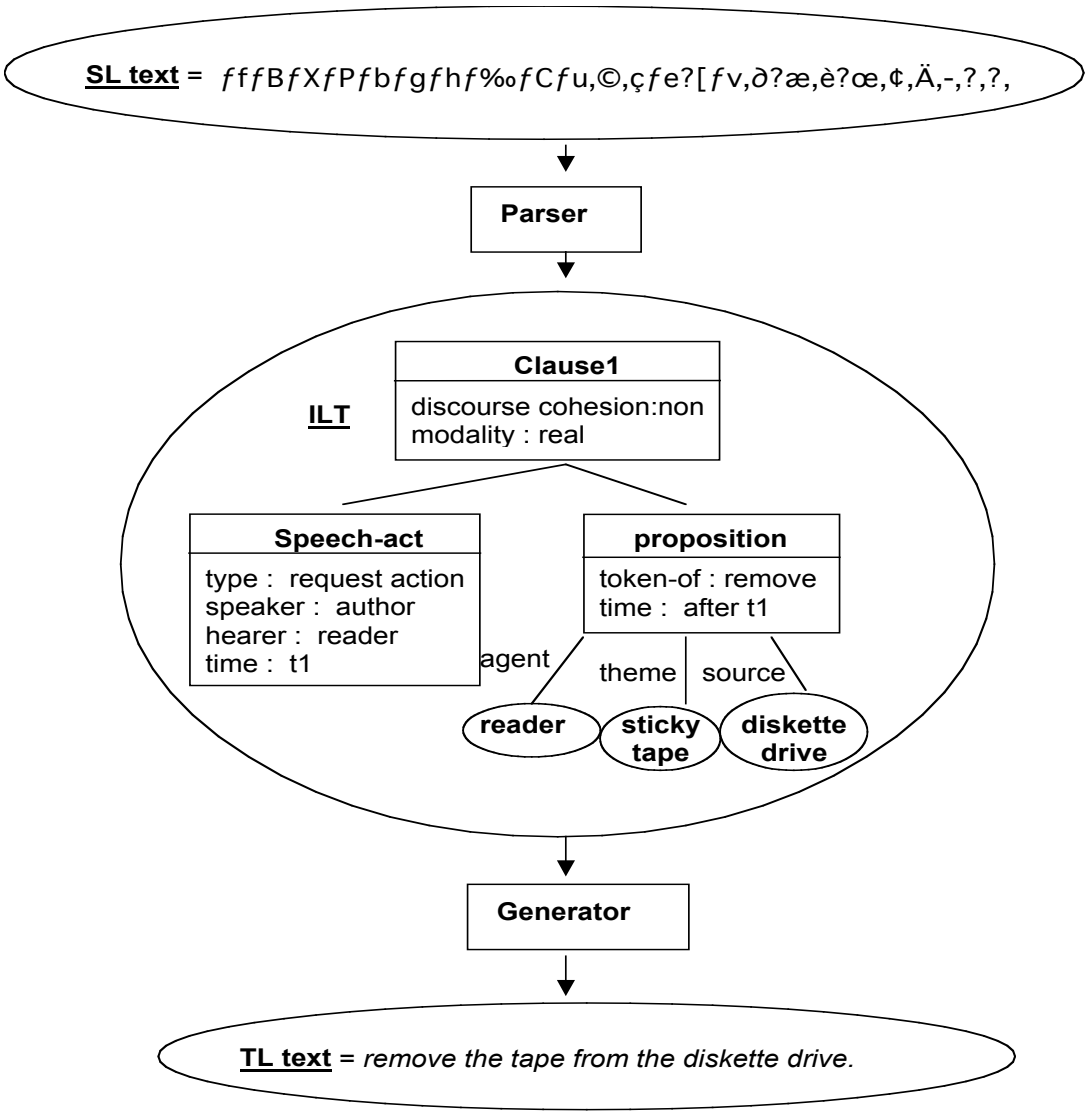


Fig. B-9 Procédure de traduction du système KBMT-89

1.2.6.3 Exemples du pivot

Nous donnons ensuite deux exemples d'interlingua, l'un venant de l'analyse d'un énoncé anglais, l'autre du japonais. Les sous-spécifications dans l'interlingua sont visibles. Un autre exemple plus détaillé avec plusieurs propositions et commentaires est donné dans l'Annexe H.

exemple 1

Dans cet exemple, l'énoncé contient 1 proposition, 1 acte de parole, 1 rôle sémantique et il manque les informations de focus et d'agent.

```
;INCHOATIVE
;ILT for "the number changed"
;KBMT-89 report p.243
```

```
(make-frame-old clause1
      (ilt-type (value clause))
      (clauseid (value clause1))
                           ;;;;; NO FOCUS;;;;
      (propositioned (value proposition1))
      (cpeechactid (value speech-act1)))

(make-frame-old proposition1
      (lit-type (value proposition))
      (propositioned (value proposition1))
      (clauseid (value clause1))
      (is-token-of (value *change))
                                  ;;;; NO AGENT ;;;;
      (THEME (value role1))
      (time (before time1)))
)
(make-frame-old role1
      (ilt-type (value role))
      (roleid (value role1))
      (clauseid (value clause1))
      (is-token-of (value *number))
      (reference (value indefinite))
)
```

Exemple 2

Dans cet exemple, l'énoncé contient 1 proposition, 1 acte de parole, 2 rôles sémantiques, et il manque les informations d'agent.

```
;JAPANESE
;"teimen –niha(wa) asi ga tuite i masu"


   « _____ »

;" a leg is attached to the bottom."
;KBMT-89 report p.244

(make-frame-old clause1
      (ilt-type (value clause))
      (clauseid (value clause1))
      (propositioned (value proposition1))
      (cpeechactid (value speech-act1)))

(make-frame-old proposition1
      (ilt-type (value proposition))
      (propositionid (value proposition1))
      (clauseid (value clause1))
      (is-token-of (value *attach))
                           ;;;;;; NO AGENT;;;;;;
      (THEME (value role1))
      (LOCATION (value role2))
      (time (before time1)))
)
```

```
(make-frame-old role1
      (ilt-type (value role))
      (roleid (value role1))
      (clauseid (value clause1))
      (is-token-of (value *artifact-leg))
      (reference (value indefinite))
)
(make-frame-old role2
      (ilt-type (value role))
      (roleid (value role2))
      (clauseid (value clause1))
      (is-token-of (value *bottom-of-3d))
      (reference (value definite))
)
```

Plus tard, dans le projet KANT, l'exploitation des cadres pour exprimer la connaissance et le pivot n'ont pas changé, mais la forme pivot est devenue plus claire, compacte et lisible. Un énoncé est exprimé par un seul cadre.

Voici un exemple de pivot dans le projet KANT,

```
(*A-CONNECT
      (argument-class agent+theme)
      (mood imperative)
      (punctuation period)
      (q-modifier
            (*Q-attach_TO
                  (case
                        (*K-TO))
                  (object
                        (*O-PHONE-LINE
                        (attribute
                              (*P-DIFFERENT
                              (degree positive)))
                        (number singular)
                        (reference indefinite)))
                  (role attach)))
            (tense present)
            (theme
                  (*O-PRODUCT
                  (number sigular)
                  (reference definite)))

Anglais: "Connect the product to a different phone line"
Espagnl: "Conecte la unidad a una linea telefonica diferente"
```

Quant au dictionnaire, voici un exemple de l'entrée « find » dans le lexique de KANT.

```
•(find
-(make-frame
-+find-v1
-(CAT (value v))
-(STUFF
-(DEFN "to discover by chance, to come across")
```

```
-(EXAMPLES "found X in the bedroom", "found X sleeping
upstairs", "found that X was sleeping at home")
-(MORPH
•     (IRREG (*v+past* found) (*v+past-part* found))
-(SYN-STRUC
      *OR*   ((root $var0)
                  »(subj (root $var1)(cat N))
                  »(obj (root $var2)(cat N))
            -((root $var0)
                  »(subj (root $var1)(cat N))
                  »(xcomp (root $var2)(cat N)(form pres-part)))
            -((root $var0)
                  »(subj (root $var1)(cat N))
                  »(comp (root $var2)(cat V)(form fin)))))
-(SEM
•     (LEX-MAP
            -(%involuntary-perceptual-event
                  »(experiencer (value ^$var1))
                  »(theme (value ^$var2))))))
```

1.2.6.4 Remarque

KANT a montré que la haute qualité de TA « fondée sur la connaissance » (KBMT) est possible quand le modèle du domaine est bien construit. Le problème du paradigme de KBMT est que la construction de cette KB (ontologie, modèle du domaine) est très coûteuse, car elle reste toujours construite manuellement [Mitamura 97]. Le défi est donc d'automatiser l'acquisition de connaissances et d'étendre la couverture des domaines.

[Czuba 98] présente un test conduit pour évaluer la portabilité du pivot KANT. Quelques phrases hors du domaine technique ont été codées et traduites. Le résultat a été satisfaisant mais l'expérimentation était trop petite pour conclure sur la portabilité vers un domaine général.

1.2.7 IF dans les projets C-STAR et NESPOLE! (1996-) (Interlingua spécialisé)

[Besacier 01] [Blanchon 00] [Levin 98, 02, 03]

1.2.7.1 Historique du système

Le projet C-STAR (Consortium for Speech Translation Advanced Research) [C-STAR] [C-STAR II] est une coopération internationale. Le thème du projet est la traduction automatique de parole dans le domaine du tourisme (dialogue client-agent de voyage), en vidéoconférence. Lancé en 1989, C-STAR I traitait 3 langues (anglais, allemand et japonais) et a effectué les premières démonstrations transatlantiques trilingues en janvier 1993. C-STAR II a pris le relais, de 1993 à 1999, en s'étendant à 3 autres langues (coréen, italien, français).

C-STAR II a présenté des démonstrations bilingues, trilingues et quadrilingues en juillet et septembre 1999. En particulier, on a pu démontrer du français-coréen, grâce à la technique du « pivot IF », alors qu'aucune des 2 équipes ne connaissait la langue

de l'autre. C-STAR III continue, avec les mêmes langues plus le chinois, et doit se terminer en 2005 ou 2006.

NESPOLE![7] (NEgotiating through SPOken Language in E-commerce) [NESPOLE!] est un projet d 30 mois qui a été financé par l'UE et la NSF (National Science Fondation) de 2000 à 2002; son but est d'explorer les futures applications de la traduction automatique de parole dans le domaine du e-commerce et du e-service. Le projet ne visait pas seulement une traduction orale viable mais aussi une investigation de la capacité de connecter deux humains ne parlant pas la même langue pour communiquer des idées, et résoudre des problèmes ensemble.

Les participants de Nespole ! étaient trois laboratoires européens (CLIPS de l'UJF à Grenoble en France, ISL de l'université Karlsruhe en Allemagne, IRST de Trente en Italie), un laboratoire américain (ISL de CMU, Pittsburgh) et deux partenaires industriels (le bureau de tourisme de Trentino et Aethra, une compagnie italienne de télécommunications). Le système prototype construit dans ce projet avait pour but de fournir une communication efficace de parole entre toutes les paires de ces quatre langues : italien, anglais, français, et allemand. Les domaines traités ont été le tourisme, les renseignements de voyage, et (un peu) les consultations médicales. Le projet a commencé en janvier 2000 et s'est terminé en juin 2002.

La structure de NESPOLE!, qu'on peut voir dans la figure ci-dessous, est basée sur l'intégration des modules que les partenaires ont développés dans C-STAR. Le médiateur (mediator) s'occupe de la communication audiovisuelle entre l'agent et le client. Le serveur HLT global (Human Language Technology) de Nespole! s'occupe de la traduction de parole capturée par le médiateur. Le serveur HLT global est constitué par les serveurs HLT spécifiques de chaque langue participante, et donc chaque HLT s'occupe de l'analyse et de la génération entre sa langue et l'IF. Il y a un « Communication Switch (CS) » qui s'occupe de la transmission de l'IF vers les serveurs HLT spécifiques.

[7] « Nespole ! »= « nèfles » en italien. « Nespole ! » signifie donc aussi « des nèfles ! », amusant jeu de mots des partenaires italiens.

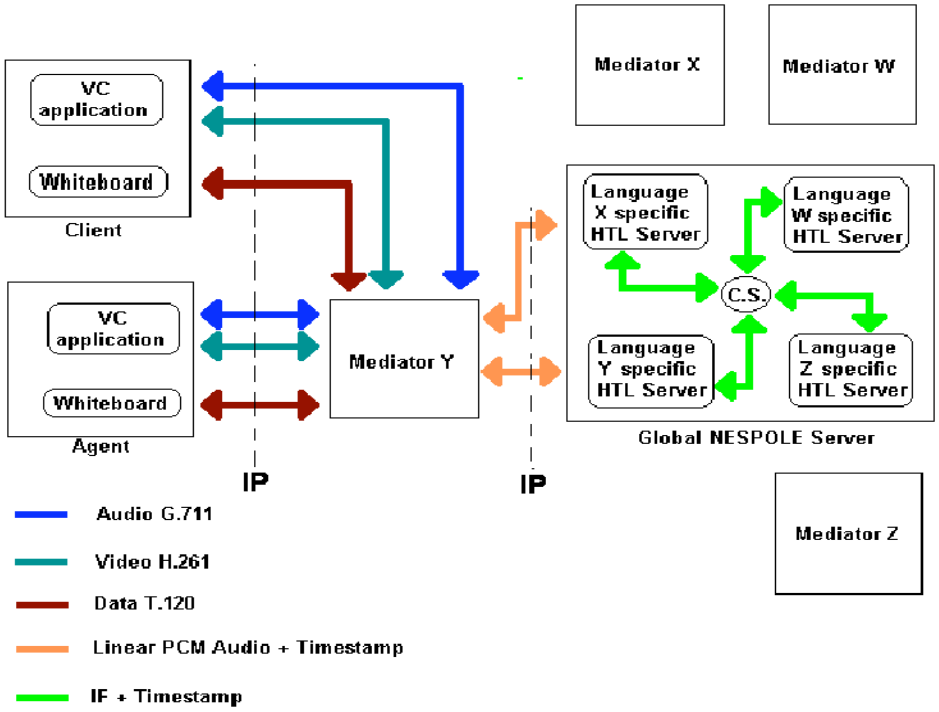


Fig. B-10 Structure de Nespole!

Voici maintenant la structure d'un serveur HLT (Human Language Techology) spécifique d'une langue.

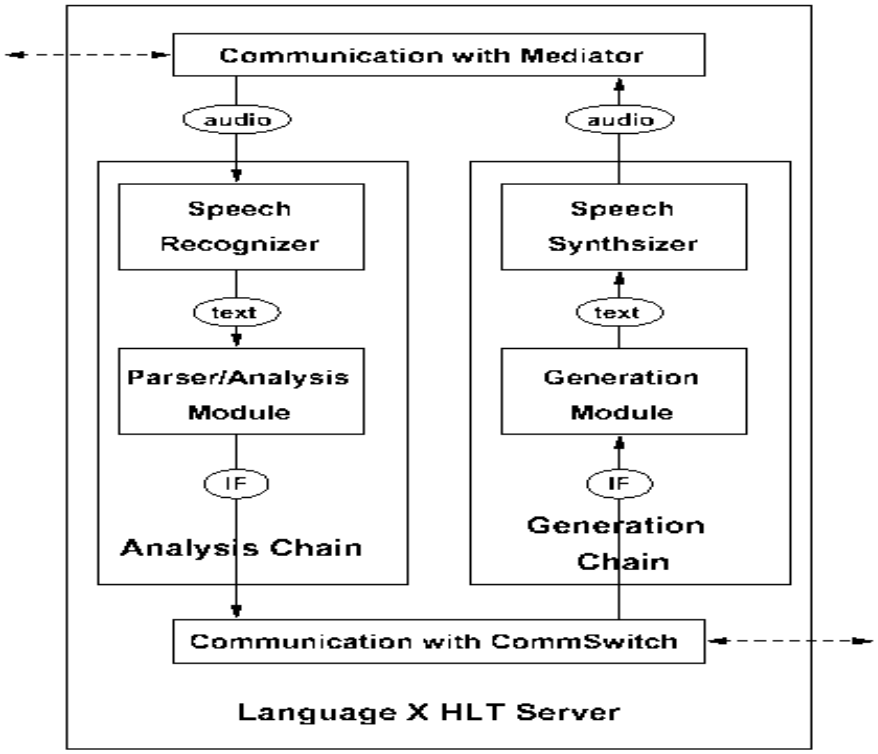


Fig. B-11 Serveur HLT spécifique de Nespole!

1.2.7.2 Description du pivot

L'IF (Interchange Format) est le langage pivot utilisé dans les projets C-STAR et NESPOLE!, avec de petites modifications dans chaque projet. Proposé par Hans-Ulrich Block de Siemens, l'IF a été adopté par le projet C-STAR II en mai 1997, et développé pendant les deux années suivantes, pour la démonstration « princeps » internationale et quintilingue (anglais, allemand, coréen, français, et italien) du 22/07/1999. Il a ensuite été exploité et très souvent modifié par le projet NESPOLE!

Le domaine sémantique de cet IF est le voyage et le tourisme, y compris la réservation et le paiement pour les hôtels, les excursions et les transports. La différence entre les IF de C-STAR II et de NESPOLE! est que NESPOLE! n'inclut pas la réservation ni le paiement, mais comprend plus de détails pour les enquêtes des hôtels, des infrastructures pour les vacances d'été et de ski à Val di Fiemme en Italie.

Le principe de l'IF est toujours le même.

L'IF est fondé sur des actes de dialogue (DA, Dialogue Act) auxquels sont adjoints des arguments. Un acte de dialogue est constitué d'un acte de parole (SA, Speech Act) complété par des concepts. Les actes de dialogue décrivent les intentions, les besoins de celui qui parle (give-information, introduce-self,…). Les concepts précisent à propos de quoi l'acte de dialogue est exprimé (price, room, activity, …). Les arguments permettent d'instancier les valeurs des variables du discours (room-spec, time, price, …).

Au moment de l'analyse, un tour de parole est d'abord découpé en « unités sémantiques de dialogue » (SDU, semantic dialogue unit). SDU est l'unité maximum du texte à analyser qui peut être représentée par un IF, et donc un énoncé d'un dialogue en langue naturelle peut correspondre à une ou plusieurs SDU.

Voici un exemple de SDU :

> anglais : «client : I would like to make a hotel reservation for the fourth through the seventh of July »
> français : « client : Je voudrais réserver un hôtel du 4 juillet au 7 juillet »
> IF : c :request-action+reservation+temporal+hotel (time=(start-time=md4, end-time=(md7, july)))

Dans cet exemple, l'étiquette « c » indique que c'est le client qui parle. « request-action » est l'acte de parole, puis il y a trois concepts : « reservation+temporal+hotel » et l'acte de dialogue est « request-action ». « time » est l'argument supérieur qui comprend deux arguments inférieurs : « start-time » et « end-time », dont « md4 » et « me7, july » sont les valeurs.

1.2.7.3 Construction et validation de la spécification de l'IF

On part de corpus de parole enregistrés et transcrits. On les « étiquette » par des énoncés IF, et on valide ensuite par examen manuel, et génération automatique. On peut aussi faire des tests avec les analyseurs mais il est difficile de déterminer automatiquement si deux IF sont synonymes, et très lourd de le faire manuellement.

La base de données officielle de C-STAR II comprend 2278 phrases anglaises et 7148 phrases non-anglaises (japonaises, italiennes, coréennes, allemand et très peu de français) [Levin 02]. Il y a 44 actes de parole, 93 concepts, et 117 arguments

possibles. En revanche, NESPOLE! comprend 65 actes de parole et 110 concepts. En plus, NESPOLE! a un formalisme de spécification permettant de définir les combinaisons légales des actions et de leurs arguments.

Nous reprenons l'exemple [Levin 02] ci-dessus pour comparer les IF de C-STAR et de NESPOLE!

« I would like to make a hotel reservation for the fourth through the seventh of July »
« Je voudrais réserver un hôtel du 4 juillet au 7 juillet »

C-STAR II

```
c :request-action+reservation+temporal+hotel (time=(start-
time=md4, end-time=(md7, july)))
```

NESPOLE!-

```
c :give-information+disposition=(who=i, desire), reservation-
spec=(reservation, identifiability=no), accommodation-
spec=hotel, object-time=(start-time=(md=4), end-time=(md=7,
month=7, incl-excl=inclusive)))
```

La différence principale est l'usage des différents actes de parole, des différents concepts et de leurs différentes compositions de variables.

1.2.7.4 Exemples du pivot « IF »

Voici quelques phrases ou énoncés que nous avons tirés du corpus de la démonstration du projet C-STAR faite le 24/09/1999 à Genève (INFOCOM). Remarquons qu'une phrase dans la conversation peut correspondre à une ou plusieurs unités sémantiques de dialogue. Et les mêmes unités sémantiques de dialogue peuvent produire des phrases différentes (comme <CLIENT :01> et <CLIENT :02>).

phrases françaises	IF
<CLIENT :01> Bonjour je suis monsieur Blanchon et c'est pour organiser un voyage à Pittsburgh.	c : greeting c : introduce-self(person-name=blanchon) c : introduce-topic+features+trip(location=pittsburgh)
<CLIENT :02> Bonjour je suis monsieur Blanchon et je veux préparer mon séjour à Pittsburgh	c : greeting c : introduce-self(person-name=blanchon) c : introduce-topic+features+trip(location=pittsburgh)
<CLIENT :03> Il me faut des billets d'avion de l'hôtellerie et je veux faire un peu de tourisme.	c : request-action+reservation+features+flight+admission c : request-action+reservation+features+hotel c : request-action+reservation+features+sight
<CLIENT :04> Deux collègues et moi arriverons le 5 mai.	c : give-information +temporal+ arrival(who=i, with-whom=(associate, quantity=2), time=(may+md5))
<AGENT :01> D'accord je note.	a : acknowledge
<CLIENT :05> Je m'appelle Richard.	c : introduce-self (person-name=richard)
<CLIENT :06> R-I-C-H-A-R-D.	c : give-information+spelling(letters=(= [r,i, c,h,a,r,d,]))
<AGENT :02> je suis désolé.	a :apologize
<AGENT :03> non plus.	a : negate

Tableau B-2 Exempls d'IF

1.2.7.5 Remarque

Dans le projet Nespole!, on a fait beaucoup de tests sur la portabilité, la portée, et la consistance du pivot IF. Le résultat paraît assez prometteur selon Lavie. L'IF a aussi été appliqué dans le domaine des dialogues médecin-patient [Lavie 01a].

Dans la prochaine étape de C-STAR, l'IF sera plus développé pour exprimer les phrases descriptives, comme « il y a un château vieux de 300 ans », en plus des phrases d'action. Mais le nombre d'actions de domaine risque d'augmenter trop vite quand le domaine d'application s'étendra.

La limitation de l'approche d'« action de domaine » est qu'elle ne fonctionne que dans un domaine très précis et très bien défini.

1.2.8 UNL (1996-) (interlingua linguistico-sémantique général)

[Uchida 01] [UNL] [UNL fondation]

1.2.8.1 Historique du système

Le projet UNL (Universal Networking Language) a commencé en 1996 sous la direction de l'IAS (Institute of Advanced Studies) de l'Université des Nations Unies (UNU). Hiroshi Uchida (____) était le responsable du projet à l'UNU/IAS.

La motivation de ce projet est que le développement d'Internet va faciliter la transmission d'information mais va aussi aggraver le déséquilibre entre ceux qui ont accès au réseau et ceux qui ne l'ont pas, si l'obstacle de la langue ne peut pas être surmonté. Pour surmonter cet obstacle, il faut, d'après H. Uchida, avoir un système pour exprimer la connaissance sur Internet et traduire la connaissance rapidement vers les autres langues naturelles. Il ne s'agit pas de construire un système de TA, mais plutôt un système de communication multilingue à large spectre (documentation technique, aussi bien que messages personnels ou informations générales et recherche d'information).

Le projet UNL a été lancé avec un plan sur dix ans. Pour les trois premières années, la tâche principale a été d'établir les spécifications du langage UNL, de développer les « déconvertisseurs » (modules de transformation entre UNL et les langues naturelles), dictionnaires, la « base de connaissances », et aussi les outils web comme UNL-Viewer.

Techniquement, UNL est fondé sur l'interlingua le plus ambitieux qu'on ait à ce jour. Maintenant il y a 12 langues (arabe, chinois, anglais, français, hindi, italien, indonésien, japonais, portugais, russe, espagnol et thaï) et une quinzaine d'équipes qui participent au projet UNL.

L'organisation se compose de deux parties: le centre UNL et les centres de langues. Le centre UNL se situe à Tokyo et à Genève et s'occupe de publier les spécifications, de définir la KB (Knowledge Base), et de l'administration. Les centres locaux s'occupent du développement des modules concernant leur langue. En 2001, la fondation UNDL a été établie à Genève pour la gestion financière et administrative ; et le centre UNL continue à s'occuper des détails techniques.

Le pivot d'UNL est un langage de graphes « linguistico-sémantiques ». Toutes les langues naturelles dans le projet doivent être exprimées en graphes UNL avant d'être

traduites vers les autres langues. Pour se distinguer des projets de TA multilingues, dans le projet UNL la transformation d'un graphe UNL en un énoncé en une langue naturelle est appelée « déconversion » et la transformation inverse est appelée « enconversion ».

Voici un exemple :

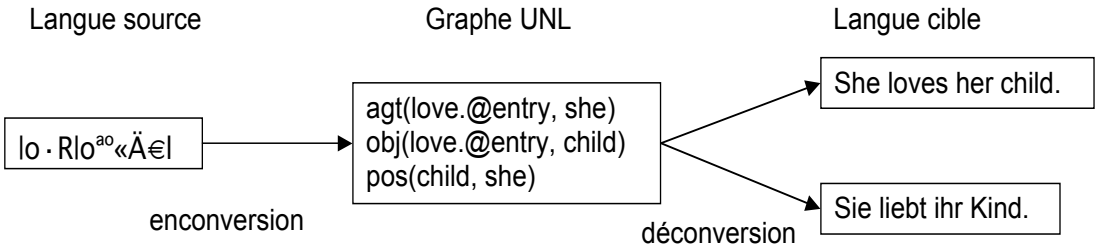


Fig. B-12 Enconversion et déconversion avec UNL

Une des différences entre le projet UNL et les projets de TA avec pivot interlingue est qu'UNL est spécialement conçu pour l'environnement Internet. H. Uchida affiche même l'ambition d'utiliser le langage UNL comme représentation intermédiaire de la connaissance sur Internet.

Le projet UNL a défini un format "UNL-html" intégré à html pour des fichiers contenant un document multilingue complet aligné au niveau des énoncés, et a produit un "visualiseur" qui transforme un fichier dans ce format en autant de fichiers html que de langues, et les envoie à n'importe quel navigateur web. Autour du noyau de traduction, il y a aussi des outils pour adapter le système à l'environnement Internet (proxys, etc.).

La structure du système UNL peut être décrite par la figure suivante [Uchida 01] :

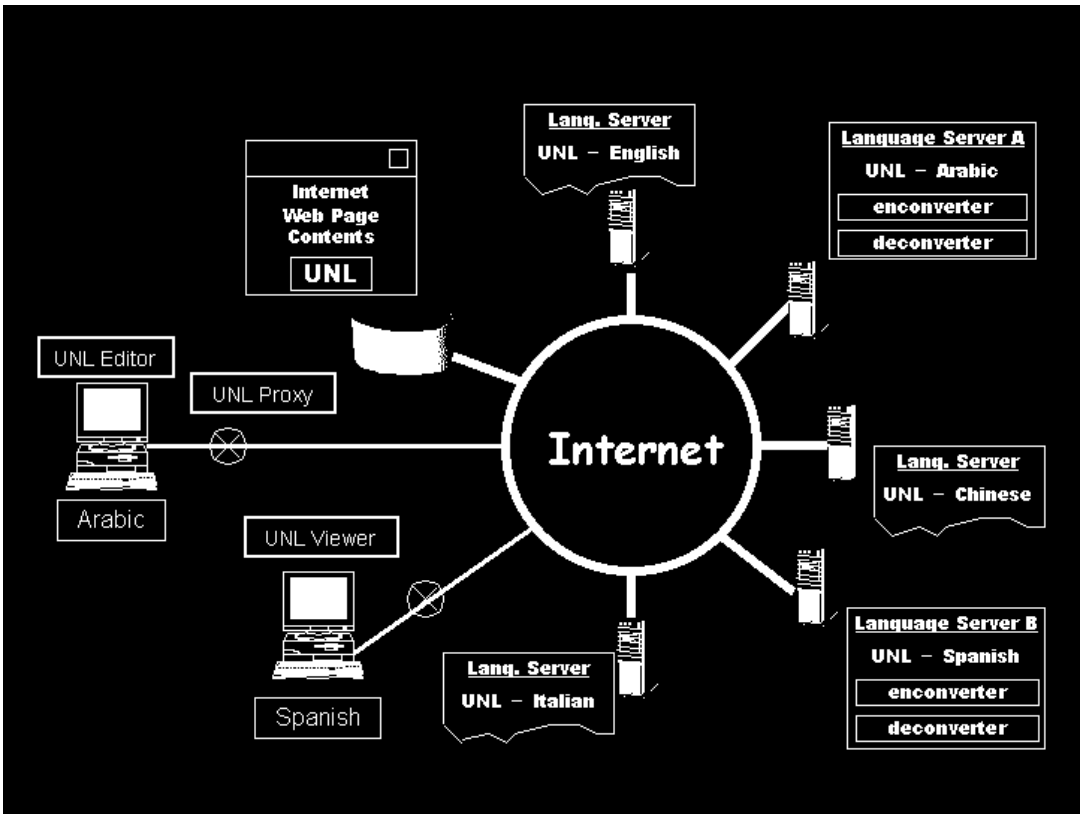


Fig. B-13 Structure du système UNL

Il y a des serveurs de langues locales qui s'occupent de l'enconversion et de la déconversion entre une langue naturelle et UNL. Un utilisateur qui a accès à Internet peut visualiser un document UNL dans sa langue (si cette version existe déjà dans ce document) à l'aide d'un visualiseur UNL, ou créer un document UNL avec l'assistance de l'éditeur UNL. Certains modules sont également téléchargeables.

Prenons la figure ci-dessus, et supposons qu'il y a un utilisateur arabe qui veut créer une page web « à la UNL ». D'abord il écrit sa page en arabe avec l'éditeur du graphe UNL, qui envoie automatiquement ce texte arabe vers le serveur arabe pour l'enconvertir en UNL. Éventuellement, le graphe UNL est amélioré par un humain travaillant sur ce serveur. Le graphe UNL est ensuite inséré dans le fichier, et une page web UNL est ainsi créée.

Si un utilisateur espagnol veut lire cette page, son visualiseur enverra le graphe UNL au serveur espagnol pour le déconvertir en espagnol, et insérer ce texte espagnol dans le document (si cela est permis, selon la gestion de document). Plus tard, si un autre utilisateur espagnol veut voir ce document, son visualiseur pourra utiliser directement le texte espagnol. Une fois qu'un document est créé « à la UNL », il peut être facilement visualisé dans d'autres langues.

1.2.8.2 Description du pivot

Le langage UNL ressemble beaucoup au pivot interlingue d'ATLAS-II et du CICC.

La représentation UNL d'un texte en langue naturelle quelconque est une liste de "graphes sémantiques" où chaque graphe exprime le sens d'un énoncé. Les nœuds contiennent chacun une unité lexicale (UW) et des attributs, et les arcs (orientés)

portent chacun une relation sémantique. Un sous-graphe connexe par arcs (en négligeant l'orientation) peut être distingué comme « portée » (« scope[8] »), de sorte qu'un graphe UNL peut être en fait un hypergraphe. Un scope est en fait un graphe replié, et non un graphe récursif, car des arcs peuvent y entrer et d'autres en sortir. Ces possibilités sont très utiles, par exemple pour représenter des constructions « à charnière » (commande) comme « Jean demande à Paul de venir le voir ».

Les unités lexicales d'UNL (UW[9]) représentent des (ensembles de) sens de mots, quelque chose de moins ambitieux que des concepts. Leurs dénotations sont construites de façon à être comprises intuitivement par des développeurs connaissant l'anglais, c'est à dire par tous les développeurs en TALN : une UW est un terme anglais ou un symbole spécial (nombre…) la plupart du temps complété par des restrictions sémantiques. Par exemple, l'UW "process" représente tous les sens de ce mot vu comme mot vedette (ici, verbe ou nom), et "process(icl>do, agt>person)" couvre seulement les sens de traiter, travailler sur, etc.

Les attributs sont le nombre (sémantique), le sexe, le temps sémantique[10], l'aspect, la modalité, etc., et les 41 relations sémantiques sont des "cas profonds" traditionnels comme l'agent, l'objet (profond), le lieu, le but, le temps, etc. Chaque graphe et chaque scope ont un unique nœud d'entrée marqué par l'attribut « .@entry » qui spécifie son centre sémantique. Une liste de restrictions et d'attributs et les spécification d'UNL sont données en Annexe A.

Une façon de voir un graphe UNL correspondant à un énoncé dans la langue L est de dire qu'il représente la structure abstraite d'un énoncé anglais équivalent "vu depuis L", c'est à dire où les attributs sémantiques non nécessairement exprimés en L peuvent être absents (par exemple, l'aspect si l'on vient du français, la détermination si l'on vient du japonais, etc.).

Voici un exemple d'une relation binaire UNL (un arc) :

```
agt(drink(icl>do).@entry.@progress, dog.@indef)

Un chien est en train de boire.
```

Dans cet exemple, deux nœuds portant les UW « drink(icl>do).@entry.@progress » et « dog.@indef », sont reliés par la relation « agt » (agent). Chaque nœud porte une UW (Universal Word), et des attributs.

Une UW est composée d'une « tête » (Headword) et d'une liste de restrictions. Ici, « drink » et « dog » sont deux têtes. « drink » a une restriction « (icl>do) » pour préciser qu'il s'agit du sens du verbe d'action. Enfin, chaque attribut est précédé de « .@ ». Ici « .@progress » exprime l'aspect progressif de l'action.

Pour construire un graphe UNL, il faut choisir des UW pour représenter les sens des mots, et les relier de façon cohérente. La KB (Knowledge Base) sert à définir

---

[8] Nous reprenons en français ce mot anglais provenant directement du grec et le prononçons avec un « o » ouvert (comme dans « télescope » ou fermé (comme dans « polygone »), au choix. Plus de détails sur les scopes sont donnés dans la section B.2.3.1.

[9] Universal Word, ou « Unité de Vocabulaire Virtuel ».

[10] *Time* par opposition à *Tense*

l'ensemble des UW et les relations possibles entre deux UW. Bien qu'une UW représente en général un ensemble de sens, on l'appelle souvent « concept » par abus de langage.

Voici un exemple complet de graphe UNL, avec les énoncés correspondants en français et en anglais, et la représentation graphique.

```
{unl}
agt(regret(icl>do).@entry, he)
obj(regret(icl>do).@entry, :01)
agt:01(come(agt>human,gol>place).@entry.@future.@not, you)
and(regret(icl>do).@entry, know(agt>human,icl>event))
agt(know(agt>human,icl>event), he)
obj(know(agt>human,icl>event), :01)
{/unl}
anglais:"He knows that you will not come and he regrets it."
français :« il sait que tu ne viendras pas et il le
regrette. »
```

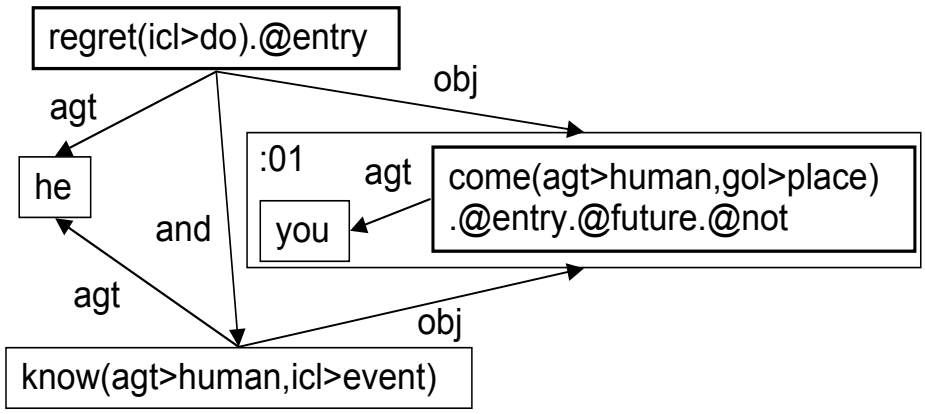


Fig. B-14 Exemple d'un graphe UNL complet

La KB définit aussi une hiérarchie entre ces concepts. Les concepts appartiennent à l'une des catégories suivantes :

- concept verbal, noté par la restriction (icl>do)
- concept nominal, noté par la restriction (icl>thing)
- concept adjectival, noté par la restriction (mod<thing)
- concept adverbial, note par la restriction (icl>how)

Cette hiérarchie est définie par 3 relations : « icl (included » définit la relation d'inclusion des concepts, « iof (instance of ») définit la relation d'instance, « equ (equal to) » définit la relation de synonymie.

Bien sûr, il est impossible de définir en extension toutes les relations entre toutes les paires de concepts. On profite de la hiérarchie de la KB pour réduire le nombre des relations. Les concepts héritent des caractéristiques de ceux placés plus haut ; les concepts du haut peuvent éventuellement remplacer les concepts du bas.

Les relations possibles entre deux UW sont définies dans la MD (Master Definition).

Voici une figure qui explique la fenêtre de la MD [Uchida 01] :

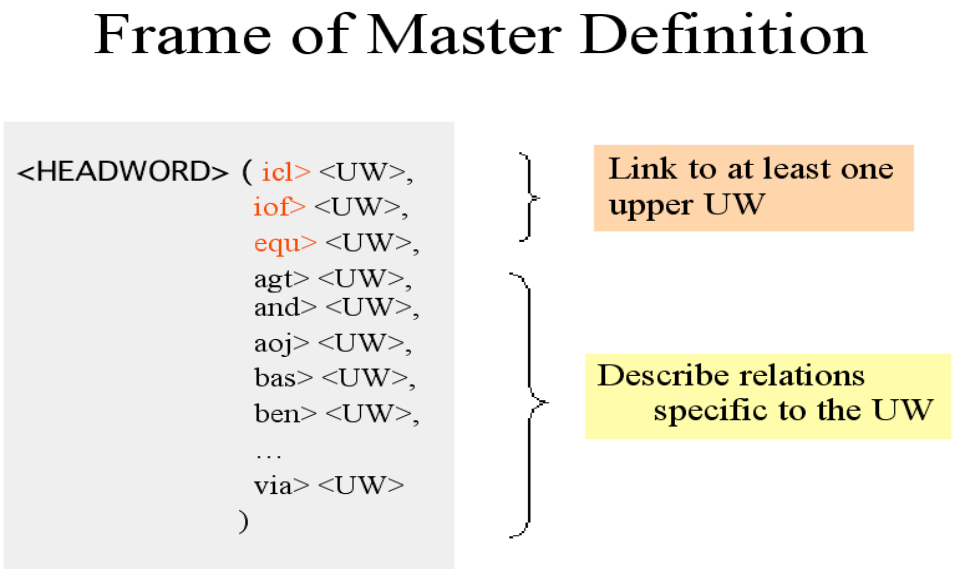


Fig. B-15 Cadre de « Master Definition »

Nous utilisons la figure suivante pour expliquer la relation entre la KB et la MD : dans la hiérarchie de la KB, le sommet est « UW » (Universal Word), qui domine quatre catégories. Les concepts nominaux héritent automatiquement la MD de « UW » (MD1), et ont leur propre MD (ici MD2). Donc la MD des concepts nominaux est en fait ({MD1} MD2). Les accolades signifient que MD1 est facultative, puisque « noun concept » est fils de « UW ». De même, UW2 et UW1 sont tous descendants des concepts nominaux, et donc ils héritent MD1 de UW et MD2 des concepts nominaux. En plus, UW2 hérite aussi la MD3 de UW1, et sa MD est donc ({MD1+MD2+MD3} MD4).

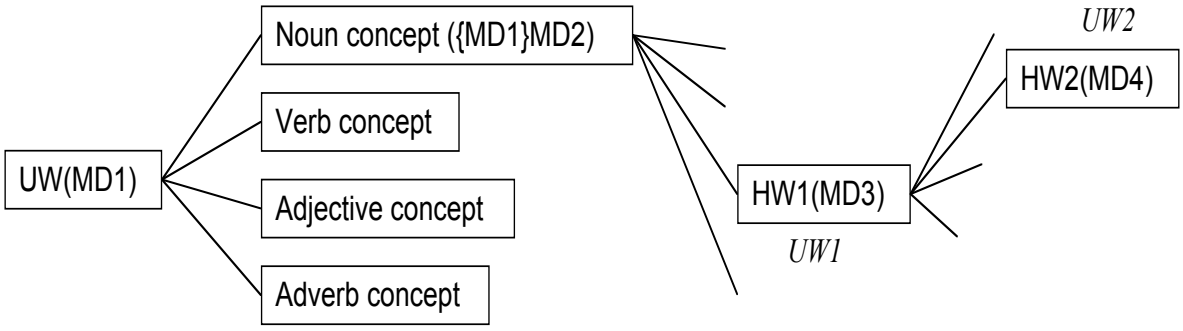


Fig. B-16 Héritage de «Master Definition »

Pour les spécifications de la KB et des MD, voir l'Annexe A.

1.2.8.3 Exemples du pivot

Il y a deux formes d'écriture linéaire du graphe UNL : tableau et liste.

Voici un exemple sur un graphe signifiant :

```
en anglais : "I can hear a dog barking outside."
en français : « Je peux entendre un chien aboyer dehors. »
```

(I) forme tableau

```
{unl}
aoj(hear(icl>perceive(agt>thing,obj>thing)).@entry.@ability,
I)
obj(hear(icl>perceive(agt>thing,obj>thing)).@entry.@ability,
:01)
agt:01(bark(agt>dog).@entry, dog(icl>mammal))
plc:01(bark(agt>dog).@entry, outside(icl>place))
{/unl}
```

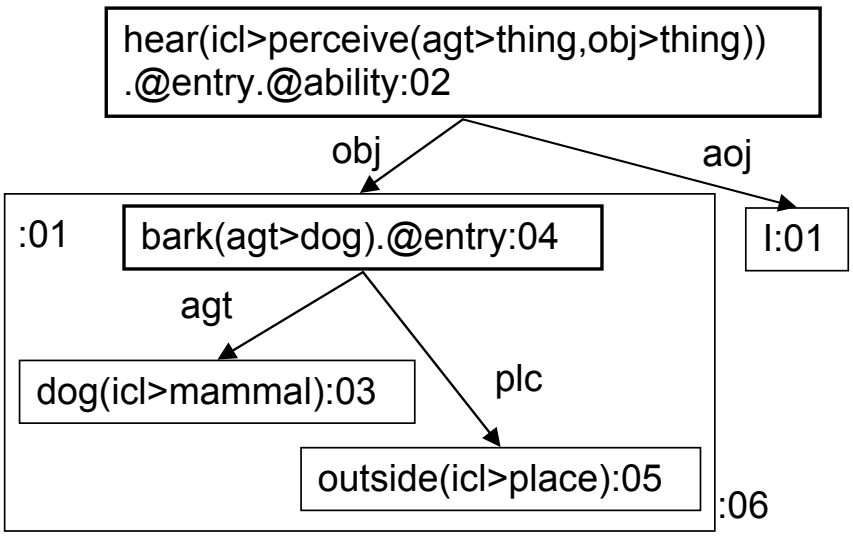


Fig. B-17 Représentation graphique d'un graphe UNL

(II) forme liste (nous marquons dans la Fig. B-17 le numéro de chaque nœud)

```
{unl}
[W]
I:01
hear(icl>perceive(agt>thing,obj>thing)).@entry.@ability:02
dog(icl>mammal):03
bark(agt>dog).@entry:04
outside(icl>place):05
:01:06
[/W]
[R]
02aoj01
02obj06
04agt:0103
04plc:0105
[/R]
{/unl}
```

Voici un extrait des premières lignes de la KB:

```
Universal Word
```

```
uw{(equ>Universal Word)}
adjective concept{(icl>uw)}
      uw(aoj>thing{,and>uw(aoj>thing),ben>thing,cao>thing,cnt>u
w(aoj>thing),cob>thing,con>uw(aoj>thing),con>do,con>occur,coo>
uw(aoj>thing),coo>do,coo>occur,dur>period,man>how,obj>thing,or
>uw(aoj>thing),plc>thing,plf>thing,plt>thing,rsn>uw(aoj>thing)
,rsn>do,icl>adjective concept})
            Afghan({icl>uw()aoj>thing{}})
            African({icl>uw()aoj>thing{}})
```

Bien entendu, ces deux dernières UW héritent automatiquement de toutes les caractéristiques de l'UW du concept adjectival (« adjective concept »).

Voici un autre exemple de KB qui montre sa hiérarchie :

```
phenomenon(icl>event{>abstract thing})
      accident(icl>phenomenon{>event})
      contingency(icl>accident{>phenomenon})
      aging(icl>phenomenon{>event})
      aging of population{(icl>aging>phenomenon)}
      brain death{(icl>phenomenon>event)}
      cerebral death{(icl>brain death)}
      bustle(icl>phenomenon{>event})
      change(icl>phenomenon{>event})
      circulation(icl>phenomenon{>event})
      climate(icl>phenomenon{>event})
            weather(icl>climate{>phenomenon})
                  rain(icl>weather{>climate})
                        hail(icl>rain{>weather})
                        shower(icl>rain{>weather})
                        snow(icl>rain{>weather})
      conformity(icl>phenomenon{>event})
      contact(icl>phenomenon{>event})
      contingency(icl>phenomenon{>event})
      convergence(icl>phenomenon{>event})
      current(icl>phenomenon{>event})
            electric current{(icl>current>phenomenon)}
            ocean current{(icl>current>phenomenon)}
                  East Africa Coast current{(icl>ocean current)}
                  East Australia current{(icl>ocean current)}
                  East Greenland current{(icl>ocean current)}
                  equatorial current{(icl>ocean current)}
      existence(icl>phenomenon{>event})
      explosion(icl>phenomenon{>event})
      extinction(icl>phenomenon{>event})
      impact(icl>phenomenon{>event})
      incidence(icl>phenomenon{>event})
      increment(icl>phenomenon{>event})
      life(icl>phenomenon{>event})
      light(icl>phenomenon{>event})
      logical phenomenon{(icl>phenomenon>event)}
      natural phenomenon{(icl>phenomenon>event)}
            heavenly phenomenon{(icl>natural phenomenon)}
                  day(icl>heavenly phenomenon)
                        daytime(icl>day{>heavenly phenomenon})
```

```
                evening(icl>heavenly phenomenon)
                      dusk(icl>evening{>heavenly phenomenon})
                morning(icl>heavenly phenomenon)
                      dawn(icl>morning{>heavenly phenomenon})
                      sunrise(icl>morning{>heavenly
phenomenon})
                night(icl>heavenly phenomenon)
      physical phenomenon{(icl>phenomenon>event)}
      physiological phenomenon{(icl>phenomenon>event)}
            breath(icl>physiological phenomenon)
                gasp(icl>breath{>physiological phenomenon})
                sigh(icl>breath{>physiological phenomenon})
      reappearance(icl>phenomenon{>event})
```

La KB est une partie essentielle du « système UNL ». Le centre UNL s'occupe de sa maintenance et de sa création. Les autres équipes de développement peuvent regarder la KB avec un navigateur de web par l'interface (UW Gate) fournie par le centre UNL.

1.3 Pivots candidats pour la coédition multilingue

Nous avons vu au total 8 systèmes à pivot, dont 5 à pivot interlingue. Voyons maintenant les avantages des différents types de pivot, pour déterminer le plus adapté à notre projet.

1.3.1 Une LN

L'avantage est que l'utilisateur peut comprendre facilement "le langage pivot" s'il connaît cette langue.

Mais cela n'aide pas la TA, car le problème intrinsèque d'ambiguïté pour chaque langue naturelle est très fort. Prenons l'exemple de Systran : si nous utilisons les modules de français-anglais et anglais-allemand pour faire la traduction français-allemand, le résultat est très pauvre.

Le seul projet exploitant une langue naturelle (sans « parenthèses cachées » comme dans DLT) comme pivot a été ATAMIRI [Guzman de Rojas 88]. L'idée initiale étant que l'aymara était une langue naturelle non ambiguë. Bien entendu, c'est faux, et le projet a rencontré toutes les difficultés prévisibles liées à l'ambiguïté intrinsèque de toute langue naturelle.

De toutes façons, en ce qui concerne la coédition multilingue, une langue naturelle n'est pas une candidate idéale pour être le pivot, car on n'a jamais eu la possibilité ou la capacité de coéditer deux langues naturelles, sauf pour des énoncés « à trous » comme dans Ambasaddor..

1.3.2 Une LN «balisée»

C'est l'approche de DLT. L'avantage est le même que celui de la langue naturelle.

De plus, en ajoutant des balises, on peut en fait décrire une structure « concrète » désambiguïsée.

Le problème est qu'une structure concrète reflète nécessairement la structure de surface de la langue, même si elle est « multiniveau ». Il faut donc très bien connaître la structure de surface de la langue en question pour pouvoir utiliser cette structure. Or ce n'est pas le cas pour la plupart des développeurs.

1.3.3 Interlingua spécialisé

Ce type de pivot peut être très précis, puisqu'il est conçu pour exprimer des concepts d'un domaine restreint. Il profite aussi des spécificités des énoncés relatifs aux tâches envisagées dans ce domaine. Il convient pour des domaines assez restreints, mais pas pour le domaine général.

Prenons l'exemple du langage IF dans le projet NESPOLE!: il encode beaucoup d'actions du domaine. Il y a beaucoup de connaissances du domaine codées et sous-entendues dans cet interlingua. Mais, si on passe au domaine général, et à des énoncés plus variés, on n'arrive plus à représenter les énoncés dans un tel IF. Selon [Boitet 01], l'expérience de l'extension d'IF en C-STAR II au domaine général a été un échec. Il y a eu des dizaines de changements de spécifications, et on avait des représentations ambiguës et malgré tout incomplètes.

Chaque domaine a ses caractéristiques. Par exemple, dans le domaine des manuels de maintenance, l'expression temporelle est presque toute le temps absente, car il s'agit de l'expression de connaissances objectives. Par contre, dans le domaine de la réservation d'hôtel, il y a beaucoup de formes de politesse, et la connaissance subjective joue un rôle important. Il est difficile de passer à un domaine trop éloigné.

Bien que la portabilité de ce genre de pivot soit possible [Lavie 01a], il s'agira toujours d'un autre domaine restreint. L'IF est un interlingua « orienté vers la tâche » (task-oriented) et il semble impossible d'étendre sa puissance descriptive au domaine général.

1.3.4 Interlingua général

Pour que le résultat de traduction soit encore plus satisfaisant, on peut coupler l'interlingua général avec un modèle du monde (KB « Knowledge Base » ou une vraie ontologie « généraliste »).

Mais un tel modèle est très difficile et coûteux à construire et à maintenir.

1.3.5 Sept critères de choix

Nous avons trouvé dans la littérature deux critères pour qu'un interlingua soit bon:

> « Un bon interlingua est fait pour exprimer non seulement ce qui est dit, mais aussi comment les choses sont dites. Donc il faut lui donner la capacité d'exprimer les connaissances subjectives » [Uchida 80].
>
> Selon [Schubert 88], un interlingua doit satisfaire au moins trois critères : autonomie (indépendance des langues naturelles), expressivité, et régularité.

Partant de là, nous proposons les sept critères suivants :

Simplicité : Si ce pivot s'utilise avec une architecture distribuée, c'est-à-dire si les sites locaux peuvent produire leur document en pivot avec une certaine consistance, ce pivot doit être facile à maintenir et à comprendre.

Généralité : Nous voulons que ce pivot ne soit pas restreint à certains domaines, mais puisse exprimer avec assez de précision des énoncés non restreints à un domaine ou une tâche particuliers.

Expressivité : Nous voulons que ce pivot soit capable d'exprimer tous les concepts des énoncés dans toutes les langues naturelles, y compris le plus possible des aspects « subjectifs » (type d'énoncé, attitude du locuteur, etc.).

Intégralité : Puisque nous voulons coéditer le texte et le pivot, il est souhaitable que le pivot puisse porter toutes les informations au niveau considéré, qu'il s'agisse de sémantique, de pragmatique, ou de référence. Si toutes les informations possibles ne sont pas présentes dans une représentation pivot, on dira qu'il est « sous-spécifié ». Par exemple, la détermination abstraite (deixis) sera souvent absente si la forme pivot résulte d'une analyse d'un énoncé dans une langue sans article (russe, japonais, chinois, thaï, etc.).

Lisibilité : Ce pivot devrait être facile à lire et à comprendre, au moins avec un entraînement minimal, et un expert devrait pouvoir produire un document dans ce pivot sans trop de peine.

Indépendance : Ce pivot devrait utiliser un vocabulaire indépendant de toutes les langues naturelles, notant l'union des « acceptions » (sens de mots) des différentes langues.

Facilité de production : Ce pivot doit pouvoir être généré automatiquement par la machine, ou au moins semi-automatiquement par des experts dans un environnement adapté.

2. Le langage UNL comme pivot pour la coédition

La discussion ci-dessus nous mène à choisir UNL comme pivot pour notre système de coédition. Nous revenons en détail sur les raisons de ce choix, puis présentons les ressources déjà développées pour UNL.

2.1 Pourquoi UNL ?

UNL est un bon pivot dans un système de coédition pour les raisons suivantes :

il est spécialement conçu pour le traitement linguistique et sémantique par ordinateur,

il a été dérivé avec beaucoup d'améliorations du langage pivot de H. Uchida utilisé dans ATLAS-II de Fujitsu, toujours évalué comme le système de TA anglais-japonais de meilleure qualité, avec une très grande couverture (plus d'un million d'entrées par langue),

les participants du projet UNL ont construit des "déconvertisseurs" d'UNL vers environ 12 langues, parmi lesquels au moins ceux allant vers l'arabe,

l'indonésien, l'italien, le français, le russe, l'espagnol et le thaï étaient accessibles pour l'expérimentation fin 2003[11],

bien qu'ils soient de nature formelle, les graphes UNL (voir ci-dessous) sont assez simples à comprendre avec peu de formation et peuvent être présentés de façon localisée à des utilisateurs "naïfs" en traduisant les symboles (relations sémantiques, attributs) et les lexèmes du langage UNL par des symboles et des lexèmes de leur langue,

le projet UNL a défini un format "UNL-html" intégré à html pour des fichiers contenant un document multilingue complet aligné au niveau des énoncés, et a produit un "visualiseur" qui transforme un fichier dans ce format en autant de fichiers html que de langues, et les envoie à n'importe quel navigateur web.

Nous montrons ensuite ce format du document « UNL-html » et nous discuterons l'exploitation de ce format plus tard dans la section B.2.4.

```
[D:dn=Mar Aral version final,on=UNL
Spain,mid=carde@opera.dia.fi.upm.es]
[P:1] [S:1]
{org:es}Yo corri ayer en el parque.{/org}
{unl}
agt(run.@entry.@past,i)
plc(run.@entry.@past,park.@def)
tim(run.@entry.@past,yesterday)
{/unl}
{cn}_____{/cn}
{de}{/de}
{el}Yesterday I ran in the park. {/el}
{es}Yo corri ayer en el parque.{/es}
{fr}J'ai couru hier dans le parc.{/fr}
[/S][/P][/D]
```

Fig. B-18 Document « UNL-html »

2.2 Ressource construites

Nous présentons maintenant les modules d'UNL construits par le centre UNL (UC, UNL Center) et par les centres de langues (LC, Language Center). Ainsi, nous aurons une vision globale de ce projet et de son environnement.

2.2.1 Pour la transformation entre la langue naturelle et le graphe UNL

Un **déconvertisseur** transforme un graphe UNL en un énoncé en langue naturelle. Les déconvertisseurs sont développés par les LC, sauf les déconvertisseurs anglais et japonais qui sont développés par UC. Les déconvertisseurs développés avec les outils de UC sont copiés sur le serveur de UC. Dans le centre UNL tournent les déonvertisseurs arabe, chinois, anglais, hindi, italien, indonésien, japonais, portugais,

[11] Pour le thaï, l'accès n'est plus public depuis début 2003, il faut demander un mot de passe.

russe (version de 2001), espagnol et thaï (version de 2002). Sur les LC, il y a les déconvertisseurs arabe, français, italien, russe, espagnol, et thaï. Il existe aussi une autre version du déconvertisseur chinois, que nous pouvons télécharger depuis le site UNL-chinois.

Le centre UNL fournit un langage spécialisé, **DeCo**, qui est indépendant de la langue naturelle et spécialement conçu pour la déconversion [UNL DeConverter 97]. DeCo se compose d'un compilateur et d'un moteur. DeCo s'occupe de la génération sémantique, syntaxique et morphologique en même temps.

Tous les centres locaux n'utilisent pas forcément DeCo. Selon [Sérasset 99], DeCo est trop simpliste pour les langues fortement fléchies comme le français et donc les centres français et russe ont utilisé leurs propres systèmes pour la déconversion. Dans [Nunes 01] il y a une explication détaillée pour la réalisation de ce module et l'écriture des règles.

Un **enconvertisseur** transforme un énoncé d'une langue naturelle en un graphe UNL. Pour l'instant, seul l'enconvertisseur de l'arabe est accessible sur Internet. D'autres (russe, espagnol, français, japonais) existent à l'état de prototypes sur les sites locaux.

Le centre UNL fournit aussi le langage spécialisé **EnCo** qui permet d'écrire des enconvertisseurs. Il peut aussi faire la désambiguïsation basé sur la KB et le contexte [Uchida 01].

UDS (UNL Development Set) est l'ensemble des outils fournis par le centre UNL pour faciliter la production de composants UNL. Il comprend DeCo, EnCo, et des outils pour construire le dictionnaire. Cela rend le projet assez modulaire. Pour ajouter une nouvelle langue dans le projet, il suffit d'écrire les règles de grammaire de cette langue en EnCo et DeCo, et le dictionnaire dans le format UNL.

Un **déconvertisseur synchrone multilingue** peut envoyer un graphe UNL à plusieurs déconvertisseurs et afficher les résultats obtenus en parallèle. Ce programme a été réalisé sous la direction de l'auteur par une étudiante pendant son stage de maîtrise [Jitkue 01]. Ce programme se trouve sur le site web SWIIVRE [SWIIVRE].

2.2.2 Pour l'intégration de la connaissance du monde réel

KB (Knowledge Base) est la « base de connaissances » du projet UNL. Ses concepteurs pensent qu'elle exprime les connaissances du monde réel, et fonctionne comme l'ontologie dans les systèmes de KBMT (Knowledge-based Machine Translation). C'est exagéré, car cette KB ne fait que définir la hiérarchie des UW, mais ne comporte absolument aucune des possibilités classiques (cadres, attributs, méthodes, typage, mécanismes d'inférence, etc.). La KB décrit aussi les relations sémantiques possibles entre 2 UW, et la hiérarchie de ces UW. Cette simplicité permet d'envisager d'atteindre une taille importante, nécessaire pour l'usage attendu, qui est la désambiguïsation. Le but de H. Uchida est d'arriver à 1,5 millions d'entrées, beaucoup plus que les 8000 concepts d'ONTOS par exemple, ou que les 6000 classes sémantiques d'ALT/JE. C'est sans doute une bonne voie, car ici la quantité prime sur la finesse de description.

La KB est accessible en lecture sur le site web du centre UNL. Voici une image de la KB.

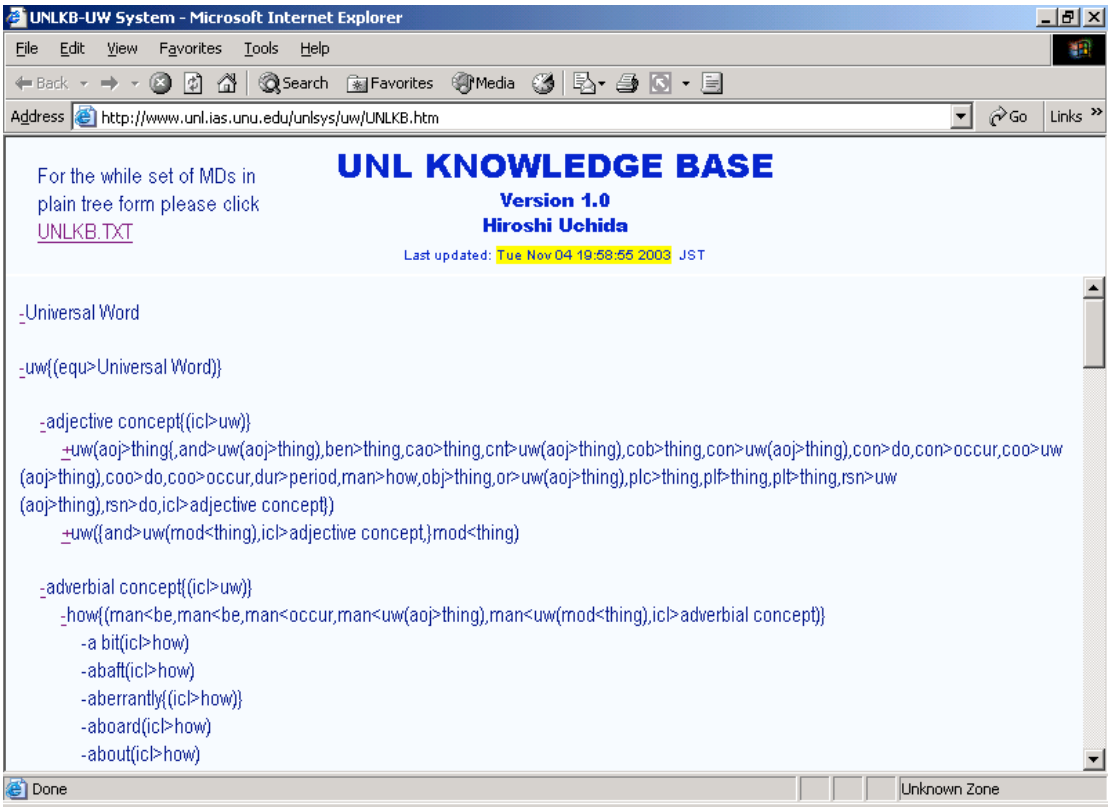


Fig. B-19 La KB présentée sur le site du centre UNL

Le **Master Dictionary** est l'ensemble des dictionnaires. Il est intégré et géré par le centre UNL. On peut le consulter soit par headword, soit par UW, ou par un mot en langue naturelle. Le résultat est l'ensemble de tous les mots dans toutes les langues naturelles qui sont liés à cette UW.

Pour chaque langue Lg, il y a au moins un **dictionnaire** UNL-Lg. On peut utiliser le **Dictionary Builder**, qui se trouve au centre UNL, pour faciliter la maintenance et la construction de ce dictionnaire.

Chaque entrée de ce dictionnaire est de la forme « [lemme-Lg]{infos-Lg}"UW";commentaire ». Voici un exemple :

« [aborder]{CAT(CATV), AUX(AVOIR), VAL1(GN)}"address(icl>do(obj>thing)"; »

Chaque ligne représente une entrée, comprenant un lemme de la langue Lg placé entre crochets, puis les informations linguistiques associées entre accolades. On trouve ensuite l'UW correspondante entre guillemets, et finalement un point-virgule éventuellement suivi d'un commentaire. Une page extraite du dictionnaire UNL-français est donnée en Annexe E.

Des copies des dictionnaires UNL-Lg se trouvent sur le site du centre UNL et dans les centres locaux. Au GETA, nous avons le dictionnaire UNL-français maître, qui comptait 38723 lemmes simples ou composés en octobre 2003, et des copies des dictionnaires UNL-Lg pour plusieurs autres langues. Nous les gérons à travers une base de données utilisable à travers le site Dicoweb [Dicoweb]

Selon les besoins des divers projets, de plus petits dictionnaires sont fabriqués. Le plus gros d'entre eux est le dictionnaire médical français-allemand-anglais-UNL qui compte 2045 entrées.

Au centre UNL, on trouve aussi, entre autres, les dictionnaires japonais-UNL avec 168766 entrées, russe-UNL avec 27484 entrées, et italien-UNL avec 21239 entrées.

2.2.3 Pour la génération du graphe UNL

Éditeur de Graphe UNL: au moins trois éditeurs ont été développés par les équipes française (trois versions), indonésienne, espagnole.

Nous montrons d'abord ici l'interface de **l'éditeur UNL de l'équipe indonésienne**. Cet éditeur a été créé à la demande du centre UNL. Il est écrit en Java. Pour créer un graphe UNL, il faut d'abord enregistrer les informations de ce graphe.

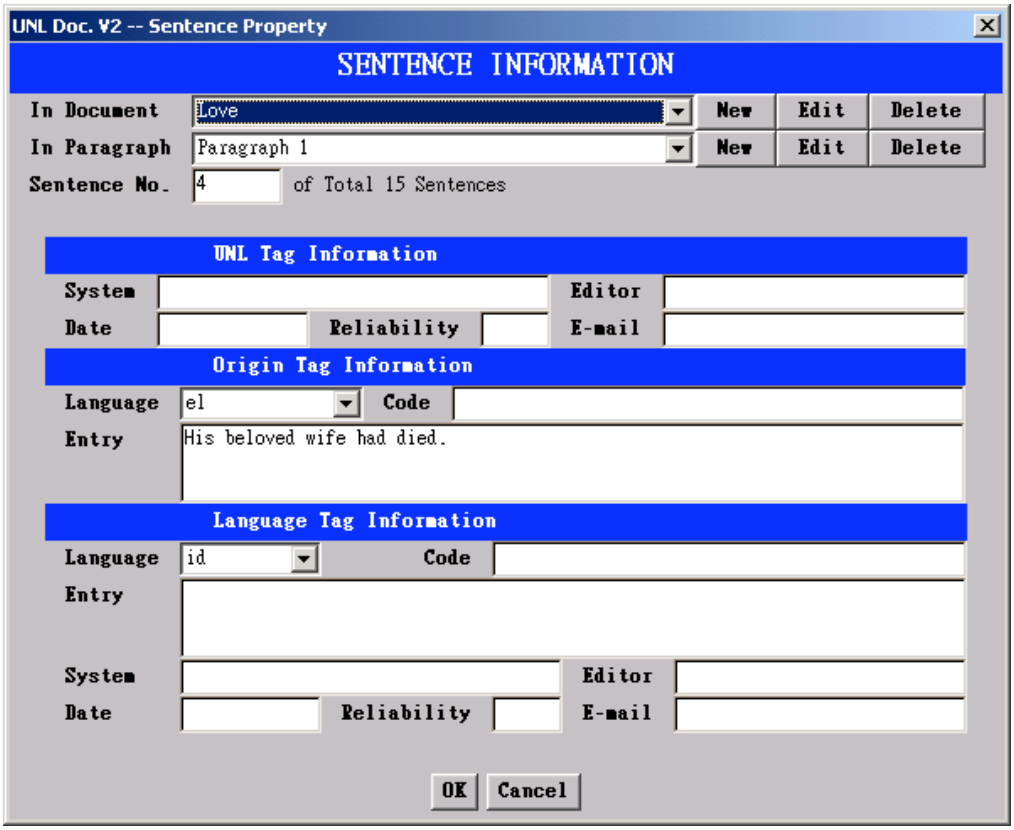


Fig. B-20 Éditeur UNL de l'équipe indonésienne (I)

Puis l'utilisateur peut cliquer sur un nœud pour éditer les informations sur ce nœud ou visualiser le graphe UNL sous forme textuelle. L'éditeur permet aussi les manipulations sur un graphe, par exemple, ajouter un sous-nœud, supprimer un graphe, etc.

Cet éditeur peut prendre un document UNL entier et naviguer dans le document phrase par phrase. La distribution de cet éditeur est limitée aux membres d'UNL.

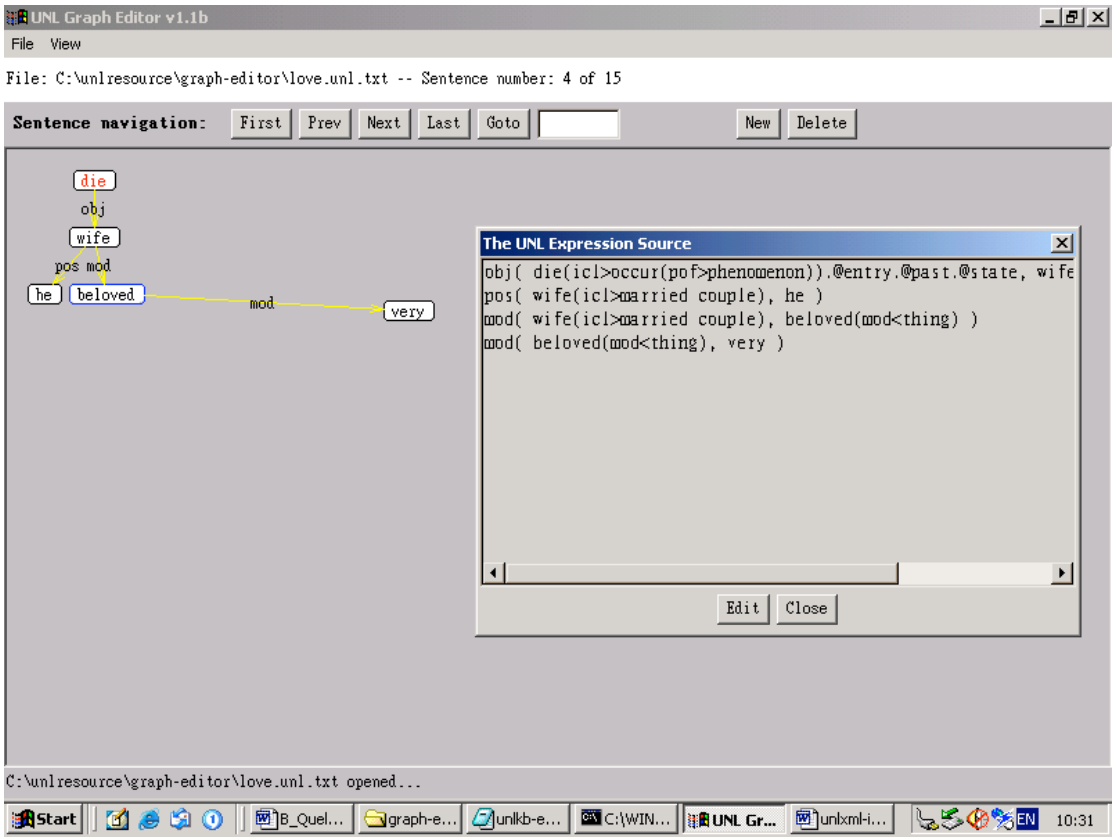


Fig. B-21 Éditeur UNL de l'équipe indonésienne (II)

A part cet éditeur, il y a aussi plusieurs autres éditeurs de l'équipe française et de l'équipe espagnole. Ils sont tous réservés à l'usage interne.

Des **corpus UNL** sont collectés sur le site SWIIVRE (voir Partie C. Section 2.1 pour plus de détails sur ces corpus).

Un **vérificateur UNL** vérifie la syntaxe d'un graphe UNL ou de tout un document. Ce programme se trouve sur le site du centre UNL [UNL]. De plus, tous les déconvertisseurs intègrent un parseur et donc un vérificateur de graphe UNL. Par exemple, les déconvertisseurs français et russe affichent un message d'erreur quand le graphe UNL entré n'est pas légal.

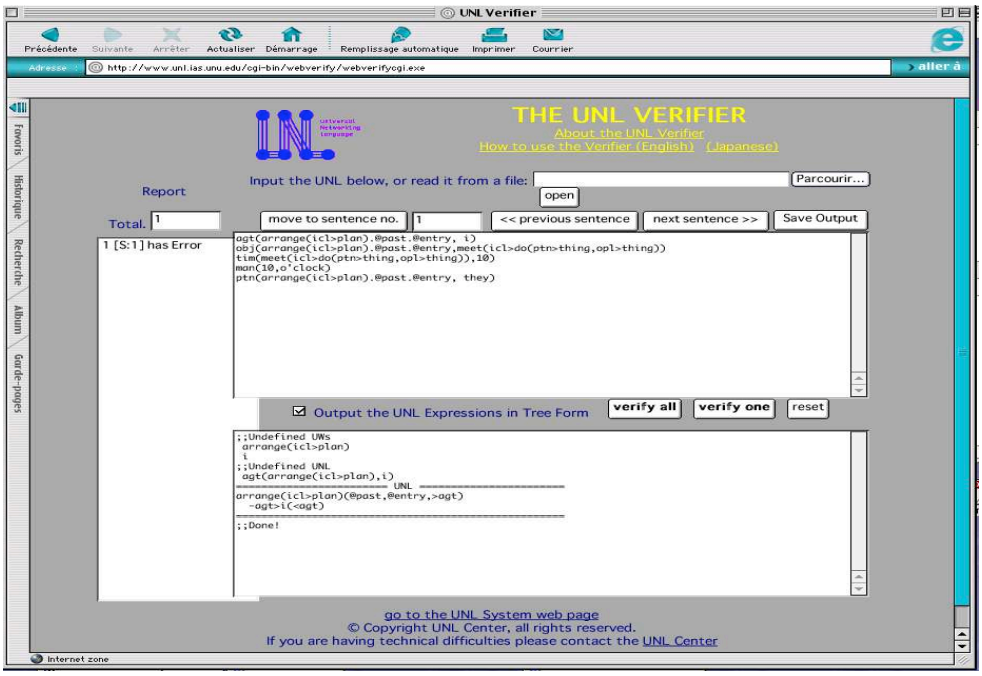


Fig. B-22 Vérificateur UNL

2.2.4 Pour l'utilisation sur le web

Nous avons construit **le site web SWIIVRE** [SWIIVRE] pour fournir des informations sur UNL et nous servir de plate-forme d'expérimentation de la coédition et de son utilisation sur le web.

L'UNL viewer du centre UNL permet de la visualisation d'un document UNL et la connexion entre l'utilisateur et les serveurs de langue. Nous discuterons plus en détail deux approches principales pour visualiser un document UNL dans la section B.2.4.2.

L'UNL proxy sert à visualiser des pages web en UNL. C'est un programme client écrit en Java installé sur un PC. Il fonctionne comme un filtre avant le navigateur pour extraire la langue spécifiée par l'utilisateur s'il s'agit d'une page web en UNL. Sinon, il transmet telle quelle la page web au navigateur. L'utilisateur peut aussi remplir les données de tous les déconvertisseurs et enconvertisseurs sur le web et l'UNL-proxy pour les contacter pour la déconversion ou l'enconversion.

Voici une figure de la structure d'UNL-Proxy [Hasan 01]:

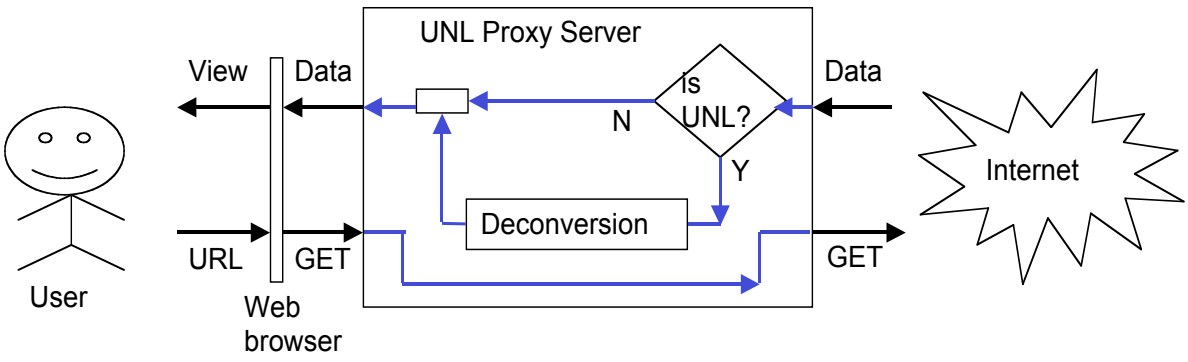


Fig. B-23 UNL proxy

Une bonne part de ces modules est conçue pour l'environnement web+Windows+PC+chercheurs du projet UNL. Ils ne sont donc pas accessibles par tout le monde depuis toutes les plates-formes. Le centre UNL ne fait pas de promotion pour ces modules et donc ils restent seulement connus entre les membres de la société UNL (UNL Society). Beaucoup d'entre eux sont conçus sans penser à l'utilisabilité et sans une maintenance continue. Il y a encore des problèmes comme le versionnage, le codage, etc. qui ne sont pas abordés. L'intégration de ces outils et la conception d'un environnement pour l'utilisateur ordinaire sont en cours.

2.3 Le langage UNL

Ici nous entrons dans le détail pour présenter ce que sont les graphes UNL et leurs avantages pour la coédition.

2.3.1 Relations, UW, scope

La base du langage UNL est la relation binaire. Une relation binaire se compose de deux UW (Universal Word) munies d'attributs et d'une relation. Une relation correspond un peu près à un cas profond (sémantique). Dans la version des spécifications la plus récente (datée du 12/2002), il y a 41 relations. A part cela, il y a aussi 3 relations réservées à la KB et à la définition de la hiérarchie des UW : icl, iof, equ. Une liste complète de toutes les relations se trouve en Annexe A.

Les attributs sont destinés à exprimer les informations subjectives[12] d'un graphe UNL. Il s'agit de la perspective du locuteur, de l'aspect ou des temps (abstraits), du mode, et aussi de l'acte de parole, de l'attitude propositionnelle, et de la valeur logique (truth value).

Dans les spécifications d'UNL (version 3 édition 1 datée du 20/02/2003), il y a 72 attributs divisés en 7 catégories :

temps (abstraits) – présent, passé, futur

aspect d'une action – achevée, inachevée, progressive, itérative, durative, fréquentative, etc.

référentielle – générique, définitive ou négative

centre d'énoncé – centre sémantique, mise en relief, titre, thème, etc.

attitude du locuteur – confirmation, ordre, politesse, interrogation, etc.

perspective du locuteur – capacité, possibilité, condition, conséquence, etc.

convention – marques de ponctuation

Une UW représente un concept simple ou un concept composé [Uchida 01]. Chaque UW se compose d'une chaîne de caractères (un mot ou terme anglais) suivie par une liste de restrictions. Il y a trois catégories d'UW : basique, restreinte, et spéciale.

[12] Terme de H. Uchida. Il s'agit de la mise en situation ou en discours du concept véhiculé par l'UW.

Une **UW basique** est représentée par un mot/un mot composé/une phrase/un énoncé anglais, qui exprime un concept proche de celui exprimé par cette chaîne de caractères en anglais. Elle peut donc être ambiguë.

Une **UW restreinte** est une UW basique désambiguïsée par une liste de restrictions. Nous trouvons les exemples suivants dans [Uchida 01] :

L'UW basique « state » peut avoir plusieurs sens, parce que le symbole « state » a plusieurs sens en anglais. Pour la désambiguïser, il suffit d'ajouter une liste de restrictions après elle, et on obtient les UW restreintes suivantes :

state(icl>do(obj>thing)) – représente le verbe « constater » en français
state(icl>nation) – représente la nation ou l'État
state(icl>situation) – représente la situation ou le stade
state(icl>government) – représente le gouvernement.

une liste de restrictions peut aussi préciser le sens d'un mot qui a un sens plus vague :

orange(icl>tree) – oranger
orange(icl>fruit) – orange
orange(icl>colour) – orangé

On utilise aussi les UW contraintes quand le symbole n'est pas ambigu en anglais, mais correspond à plusieurs mots ressentis comme de sens différents dans une autre langue, par exemple :

« marry(icl>do) » (se marier) n'est pas ambigu en anglais, mais, en russe ou en chinois, il y a deux mots selon qu'un homme ou une femme se marie. Donc il faut ajouter une restriction :

marry(agt>male) – « _____ » (russe), « _ qu3 » (chinois)
marry(agt>female) – « _____ _____ » (russe), « _ jia4 » (chinois)

Une **UW spéciale** est un moyen pour introduire des concepts qui ne se trouvent pas en anglais :

ikebana(icl>activity, obj>flower) – art floral japonais
samba(icl>dance) – genre de danse
soufflé(icl>food, pof>egg) – aliment fabriqué avec des œufs

Enfin, on peut indiquer indirectement la catégorie (sémantico-pragmatique) de l'UW, pour économiser le nombre des symboles et exprimer plus de sens :

answer(icl>do) – « do », donc prédicat (verbe « répondre », « to answer »)
answer(icl>thing) – « thing », donc entité (substantif, « réponse »)

weekly(icl>how) – par semaine
weekly(mod<thing) – hebdomadaire

positive(mod<thing) – dans « le résultat positif »

positive(aoj>thing) – dans « le résultat est positif »

La syntaxe des UW en BNF se trouve en Annexe A.

Nous avons montré la manière dont UNL peut s'adapter et inclure les concepts dans les langues autres que l'anglais, et aussi la désambiguïsation des concepts. Dans la communauté UNL, on développe des procédures pour normaliser l'usage des restrictions de façon à obtenir un système d'UW cohérent et complet. Le projet FB2004 a été un progrès en ce sens, et cela continue avec un projet « UNESCO ».

Cependant, il est inévitable que certains concepts ou groupes de concepts aient plusieurs UW synonymes créées par des groupes différents. En effet, il peut être difficile de déterminer la synonymie avec certitude, et dans un tel cas on préfère par précaution créer une nouvelle UW. Ce problème n'est pas trop grave dans la mesure où deux UW synonymes sont nécessairement très proches dans la KB.

Enfin, nous voulons donner la définition d'un « **scope** ». Un « scope » est un sous-graphe connexe par arcs constitué de tous les arcs de même numéro de « scope », et des nœuds associés. Un (et un seul) de ses nœuds doit porter l'attribut « .@entry ». Un scope est donc en fait un graphe replié et il peut avoir les caractéristiques d'un nœud (c'est-à-dire, être suivi par des attributs, être le nœud entrée, etc.) mais il est plus expressif qu'un nœud. Il faut souligner qu'un scope peut se déplier et qu'alors des arcs peuvent sortir de ce sous-graphe et aller vers l'extérieur ou que des arcs peuvent y arriver en venant de l'extérieur.

Voici un graphe UNL qui a un scope avec un arc qui sort vers l'extérieur.

{org:es}
Los ríos dejaron prácticamente de llegar, taponados por presas.
{/org}
{unl}
obj(flow(icl>occur).@past.@entry.@not, river.@def.@pl)
man(flow(icl>occur).@past.@entry.@not, almost)
rsn(flow(icl>occur).@past.@entry.@not, :01)
obj:01(block(icl>do).@past.@entry, river.@def.@pl)
agt:01(block(icl>do).@past.@entry, dam.@pl)
{/unl}
{fr}Bloquées par des barrages, les rivières ne coulaient presque plus.{/fr}
{el}Blocked by the dams, the rivers almost stopped flowing.{/el}

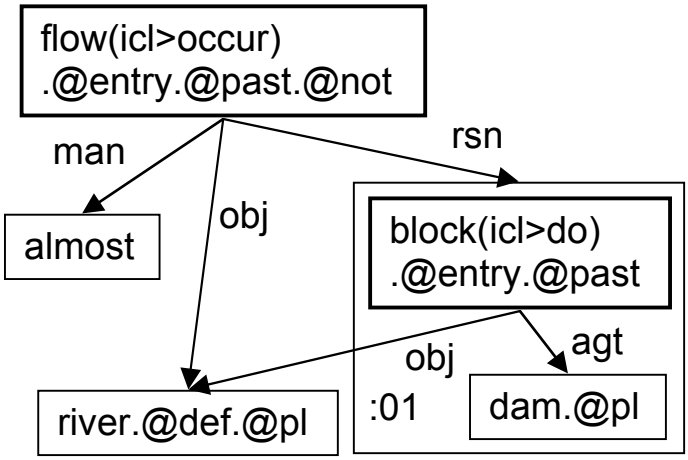


Fig. B-24- Scope avec arc allant vers l'extérieur

2.3.2 Problème de sous-spécification

Comme déjà constaté dans [Boitet 88b], le problème de sous-spécification est très connu en TA multilingue. Un projet multilingue comme UNL, qui couvre plusieurs langues très éloignées, ne peut pas y échapper.

Un graphe UNL correspond à un énoncé dans la langue L exprimé en anglais. Dans un graphe créé par un locuteur chinois, la détermination est très souvent sous-spécifiée parce qu'il n'existe pas d'article en chinois. Les nœuds correspondant aux noms dans un graphe UNL créé à partir du chinois n'auront donc le plus souvent pas d'attribut de détermination, et donc le graphe sera sous-spécifié de ce point de vue par rapport aux langues à articles.

De même, pour déconvertir en arabe, il faut l'indication du duel, absente si le graphe est produit à partir d'une langue sans duel.

Même si un graphe UNL provient d'une langue assez proche de l'anglais, il est difficile de le rendre neutre et complet. Nous avons vu des graphes UNL « à l'espagnole » et « à la française ». On constate que l'exploitation de l'article ou du pluriel dans les représentations UNL de ces langues est différente, et cela peut créer des ambiguïtés.

Pour résoudre cela, il faut d'abord établir un consensus sur les spécifications pour avoir la possibilité d'ajouter autant d'attributs que nécessaire, tout en restant dans des limites raisonnables. Mais, même si les spécifications d'UNL permettaient aux utilisateurs de spécifier tous les phénomènes de langue, les enconvertisseurs, même aidés par les utilisateurs, risqueraient de ne pas toujours bien mettre tous les renseignements dans le graphe, simplement parce qu'ils ignorent que certains renseignement sont cruciaux dans la déconversion vers d'autres langues. Il faut donc pouvoir les ajouter *a posteriori*, après la déconversion, quand les erreurs dans le texte sont constatées.

Pour l'instant, les relations et attributs ne sont pas suffisants pour exprimer tous les phénomènes de langue. En fait, on doute fortement qu'un tel interlingua complet puisse un jour exister. Mais nous pouvons ajouter des attributs pour améliorer l'expressivité d'UNL. Avec la grande couverture et la variété des langues qu'il couvre, UNL est d'ailleurs bien placé pour constater les manques et y remédier.

2.3.3 Nécessité d'une « normalisation » de la méthode de représentation des phénomènes linguistiques en UNL

Bien que les relations UNL correspondent à peu près aux rôles sémantiques, l'expérience montre qu'il est impossible d'avoir une interprétation consistante des relations argumentaires (valences logiques fortes) par ces 41 relations. Le fait qu'il y a 15 groupes de participants venant de pays différents complique encore l'interprétation de ces relations UNL.

Un comité a donc été établi en 2000 après le symposium UNL à Genève pour examiner les spécifications d'UNL et pour promouvoir un « bon encodage » en UNL. Ensuite, le projet FB2004 [FB2004] a expérimenté l'encodage proposé par une procédure d'expérimentation et de débat public.

Selon les conclusions de ces projets, publiées dans plusieurs colloques UNL [Boguslavsky 02a, 02b] [Boitet 02d], la correction et la non-ambiguïté d'une expression UNL ne suffisent pas pour une déconversion correcte vers autres langues ; on doit chercher à produire des expressions UNL adéquates. Une expression UNL adéquate doit non seulement préserver le sens du texte original, mais aussi être facile à utiliser dans les applications, y compris la déconversion vers d'autres langues.

Les problèmes principaux qui empêchent l'encodage dans une forme adéquate sont les suivants :

> Les UW composées de plusieurs mots : par exemple « International Monetary Fund », « Ministery of foreign affairs », etc. Les noms propres de ce genre sont innombrables et donc il est impossible de créer une entrée dans le dictionnaire pour chacun comme fait le centre UNL.
>
> Les verbes support : UNL utilise des symboles issus de l'anglais, mais les verbes support ne sont en général pas les mêmes dans les autres langues. Par exemple, pour exprimer « prendre une douche », en anglais « take a shower », le verbe support est « take ». Mais le verbe support en russe sera « recevoir », et en chinois « se laver ». Idem pour les verbes composés « take an action », « give a lecture », « make an impression », etc. Il est difficile de déconvertir correctement si on ne considère que le verbe lui-même. Il faut donc que l'analyse reconnaisse les cas où un verbe est verbe support, et alors le traduise dans l'UW correspondant au verbe support anglais, ou bien traduise tout le prédicat composé en une UW adéquate.
>
> Les relations prédicat-argument : les spécifications UNL ne permettent pas de spécifier les arguments d'un prédicat, ni dans le dictionnaire ni dans les graphes.
>
> Le problème de la distinction entre restrictif et non-restrictif – par exemple, dans la phrase anglaise « Wise Greek diluted the wine with water », « Greek » peut être restrictif (seulement les Grecs intelligents diluent le vin dans l'eau, les stupides non) ou non-restrictif (généralement les Grecs sont intelligents et ils diluent le vin dans l'eau). Dans les spécifications UNL, il n'y a pas de moyen de faire cette distinction. Or, c'est important en français de distinguer « les Grecs » et « des Grecs », ou encore, « les Grecs intelligents, » (épithète) et « les Grecs, intelligents, » (attribut).

Les conventions sur les attributs□on n'est pas sûr si par exemple, une UW sans marque «□@def□ a pour valeur «□ndéfini□, ou simplement si l'encodeur ou l'enconvertisseur n'a pas pu ou voulu la noter ou pu la calculer. De plus, il y a des mots anglais qui ont des sens différents au singulier et au pluriel, et dans ce cas un attribut «□@pl□ attaché à l'UW associée peut être ambigu.

Le problème des anaphores□il est important pour des langues comme le français d'avoir cette information, souvent inter phrastique, pour décider le genre des noms anaphoriques, mais elle est absente d'UNL qui n'a pas d'attribut .@eld permettant de mettre à la place de l'UW «□it□, «□he□, «□he□, l'UW référée, qu'elle soit ou non dans la même phrase.

De façon générale, comme un graphe UNL correspond à une seule phrase, les informations inter phrastiques ne peuvent pas être exprimées.

Plusieurs solutions ont aussi été proposées dans ces articles pour résoudre ces problèmes. Il faudra voir dans les projets futurs si ces propositions peuvent être respectées et améliorer de la qualité des expressions UNL.

2.3.4 Nécessité de «□normalisation□ de la procédure de l'encodage entre les équipes

2.3.4.1 Problème

En plus de la normalisation de l'expression en UNL que nous avons discutée ci-dessus, il y a aussi le développement des UW d'UNL qui retient l'attention. Il s'agit de la synchronisation des dictionnaires des différentes équipes. Chaque équipe doit mettre au point non seulement son déconvertisseur, mais aussi son dictionnaire.

En théorie, le vocabulaire UNL devrait être centralisé dans la KB, et chaque groupe devrait s'assurer que ses UW sont cohérentes avec celles des autres groupes. Mais malheureusement ce n'est pas le cas. Les entrées dans la KB n'arrivent jamais à couvrir le vocabulaire dont les équipes locales ont besoin.

Par contre, on est arrivé à une telle homogénéisation dans le sous-projet FB2004, avec seulement 5 langues, et une organisation pratique beaucoup plus efficace que celle de la KB.

2.3.4.2 Projet FB2004

Disons donc quelques mots de ce projet. FB2004 signifie « Forum Barcelona 2004 ». Dans le projet FB2004-UNL, il s'agissait de préparer un démonstration. Le projet fut lancé en avril 2001. Les textes originaux sont en espagnol et en anglais, et le contenu est l'introduction du festival écrite par le directeur de l'UNESCO. Cela constitue un corpus d'environ 2800 mots (11 pages environ). Cinq équipes ont participé à ce projet : espagnole, italienne, russe, française, et indienne.

Ce projet avait aussi pour but de développer une méthode de coopération entre les équipes.

Le projet a eu deux phases : dans la première phase, les équipes française, espagnole, italienne et indienne ont partagé l'enconversion de 30 phrases parmi les 122 phrases

du document, et l'équipe russe a été l'intermédiaire pour le débat et la discussion de l'encodage du graphe UNL. Toutes les équipes devaient aussi commenter les graphes enconvertis par les autres équipes. Un forum web a été établi pour cela. Une fois que tout le monde a été d'accord avec les graphes enconvertis, les graphes ont été déconvertis dans les 5 langues.

Deux articles écrits par Boguslavsky [Boguslavsky 01a, 01b] contenant les conseils de bon encodage et sont disponibles sur le site du projet [FB2004]. Dans la phase II, la même procédure a été appliquée aux 92 phrases restantes.

La procédure d'encodage coopératif entre les équipes a été définie comme suit [Cardeñosa 01a, 01b] :

étape 1 : soumission du code UNL.

étape 2 : soumission par courriel.

étape 3 : collecte des UW.

étape 4 : collecte des données sur les coûts d'encodage.

étape 5 : inspection du code UNL.

étape 6 : discussion sur le forum du site web du projet FB2004-UNL.

étape 7 : correction du code UNL avec le tableau d'UW de la dernière version.

étape 8 : inspection du graphe UNL avec le résultat de la déconversion.

étape 9 : collecte des données des coûts de la déconversion.

étape 10 : collecte des données des coûts de la post-édition.

L'intérêt de ce projet est que, pour la première fois, des équipes locales se sont unies pour évaluer la qualité des graphes UNL et discuter de la façon d'arriver à un bon encodage.

On a essayé d'évaluer le coût d'encodage et de déconversion, bien que le résultat ne soit pas listé sur le site. On a aussi proposé une méthode pour collecter les UW pour que la procédure d'encodage soit plus efficace.

La procédure d'unification des UW est assez simple : le fournisseur du texte original fournit une liste des UW des expressions UNL du projet avec le texte original et les graphes UNL. Dans cette liste, on spécifie les UW existantes, les UW proches mais pas identiques, et les UW qui n'existent pas encore. Les autres équipes remplissent cette liste en liant ces UW à des mots dans leurs langues.

Voici un extrait de cette liste d'UW [Cardeñosa 01a, 01b] :

Sentence numbers (sentences where the UW appears)	**Headword** (Translation to English)	**Universal word**	**Spanish Headword** (in the native language)	**French Headword** (in the native language)	**Russian Headword**	**rem ark**
93	transformation	transformation	transformación	transformation	(à remplir)	
93, 94, 97	urban	urban	urbana	urban		
93	carry out	carry_out(icl>do)	llevar a cabo	réaliser		
93	build	build(icl>do)	construir	construire		
93	site	site(icl>thing)	escenario	site		
93, 97, 98	forum	forum	fórum	forum		
93	satisfy	satisfy(icl>do)	responder a	satisfaire		
93	effectively	effectively	forma eficaz	effectivement		
93	need	need(icl>thing)	necesidades	besoin		
93, 94, 97, 99	city	city	ciudad	cité		

Tableau B-3 Table pour l'échange d'UW dans projet FB2004

Avec la normalisation de l'encodage des énoncés de langue naturelle en UNL et la normalisation de la procédure d'encodage entre les équipes, nous espérons pouvoir simplifier les calculs des centres locaux pour trouver les bons mots. A plus long terme, le centre UNL devrait reprendre cette méthode et modifier les spécifications et la procédure de travail.

A part la TA, le langage UNL a aussi été appliqué au résumé de texte [Sornlertlamvanich 01], à l'analyse de texte [Choudhary 01], et à l'annotation sémantique de lexiques [Sornlertlamvanich 00].

2.4 Formats de documents UNL et outils associés

Des le début du projet UNL, le centre UNL a défini le format UNL-html, que nous appelons ici UNL-html.1.

2.4.1 UNL-html.1 et UNL-html.2

Un document UNL-html est un document multilingue balisé basé sur html, avec des balises non-html (entre [] et {}), utilisées pour marquer les informations spécifiques à UNL. Il y a deux types de balises : les crochets [] délimitent la segmentation du document (phrase, paragraphe et document). Les accolades {} délimitent les données linguistiques comme le graphe UNL et la version de langue.

Nous distinguons deux types de format UNL-html et nous les nommons UNL-html.1 et UNL-html.2. Le format UNL-html.1 est le format défini par le centre UNL.

Voici un document UNL-html.1 visualisé sous un éditeur textuel (Notepad de Microsoft).

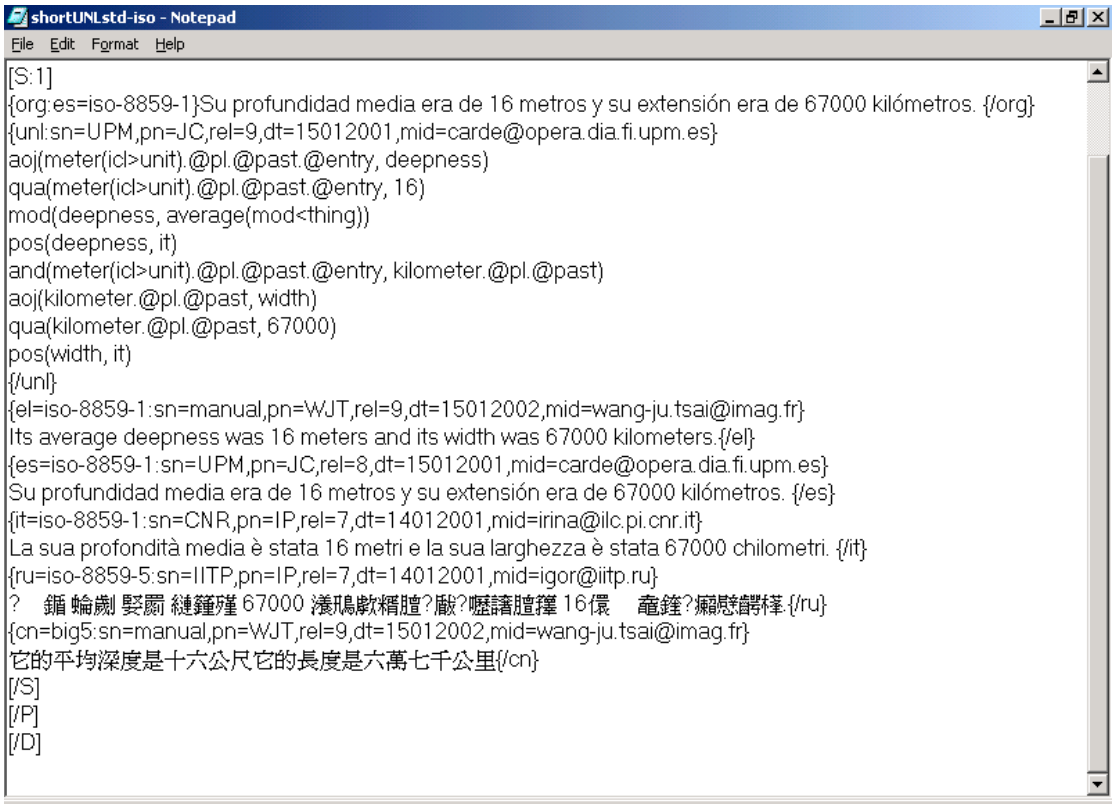


Fig. B-25 Un document UNL-html.1

Un document UNL-html a une hiérarchie arborescente comme le montre la figure (Fig. B-26). Chaque document se trouve entre les balises de document [D] et [/D]. Dans un document, il y a au moins un paragraphe et dans un paragraphe il y a au moins une phrase. Les balises de paragraphe et de phrase sont [P] [/P] et [S] [/S].

Dans une phrase, il y a un graphe UNL (éventuellement vide s'il n'a pas été construit) et le texte original, d'où ce graphe est enconverti et la/les version(s) de langue(s) déconvertie(s). Les balises sont : {unl}{/unl} pour le graphe UNL ; {org}{/org} pour le texte original. Pour les langues déconverties, on utilise des balises correspondant au code ISO à deux caractères de chaque langue. Par exemple, {ab}{/ab} encadrent un texte arabe et {cn}{/cn} un texte chinois, etc.

Dans les balises (sauf au niveau de paragraphe), on peut ajouter une liste d'attributs pour noter certaines informations, comme le nom du document, l'auteur, la date, l'adresse électronique de l'auteur, le codage du texte, etc. Syntaxiquement, les attributs sont des paires « nom=valeur » séparées par des virgules, et deux points « : » indique le début de la liste d'attributs.

Il y a deux exceptions à cette syntaxe. La première est qu'un chiffre avec deux points peut être attaché directement après P et S pour indiquer le numéro du paragraphe ou de la phrase. La deuxième est que le codage des caractères est attaché directement après l'étiquette de langue, suivie de « = ».

Voici une figure représentant la structure arborescente d'un document UNL-html.1.

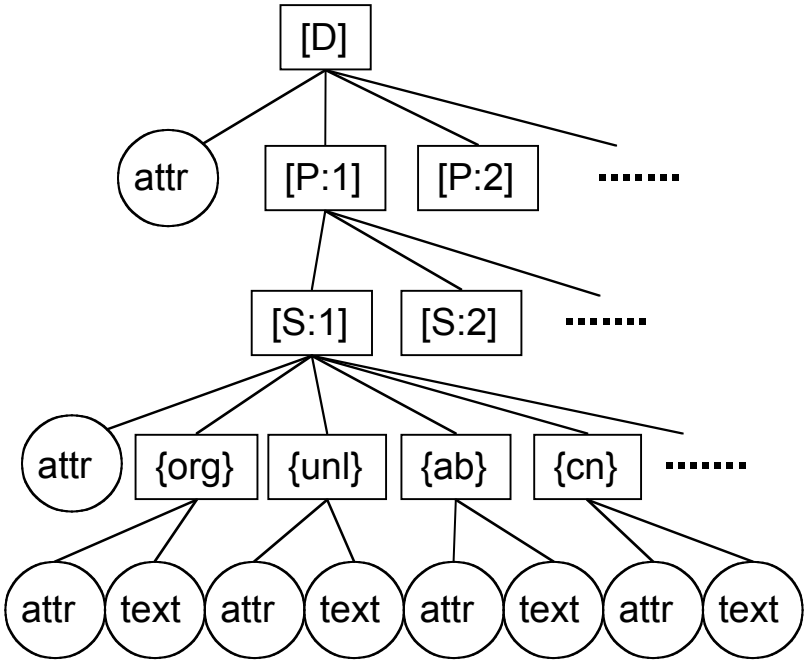


Fig. B-26 Structure d'un document UNL-html.1

Le format UNL-html.1 a été conçu avant l'avènement de XML, et ne permet pas d'utiliser directement les outils développés pour XML ; il faut à chaque fois développer une application spécifique (comme le UNL-viewer).

Ce n'est pas non plus un format à montrer à l'utilisateur. Pour lire le document dans la langue d'un utilisateur, il faut d'abord extraire du fichier UNL-html.1 un fichier html « normal » ne contenant que la langue en question. Enfin, comme les parties en différentes langues d'un document UNL-html.1 sont dans différents encodages et pas en Unicode, on ne peut pas visualiser toutes les langues à la fois (par exemple, en colonnes synchronisées) avec les éditeurs et navigateurs du commerce. Par exemple, la version russe dans la Fig. B-25 est illisible parce que la police d'éditeur de l'auteur est par défaut Big5 qui ne contient pas les caractères cyrilliques.

Cependant, si on ajoute des balises html contenant l'attribut d'encodage, on peut visualiser directement un fichier UNL-html sous un navigateur quelconque, à condition bien sûr que toutes les polices nécessaires soient installées.

Il y a donc deux formes légèrement différentes d'un document UNL-html. La première, publiée par le centre UNL, suppose que tous les caractères significatifs sont implicitement échappés. On laisse donc « mod<thing ». La deuxième est la forme réellement traitable par un processeur html. Dans notre exemple, il faut alors remplacer « mod<thing » par « mod<thing ». Nous proposons, quand ce sera nécessaire, d'appeler ces deux formes UNL-html.1 et UNL-html.2.

Voici un document UNL-html.2 sous un éditeur textuel (Notepad de Microsoft).

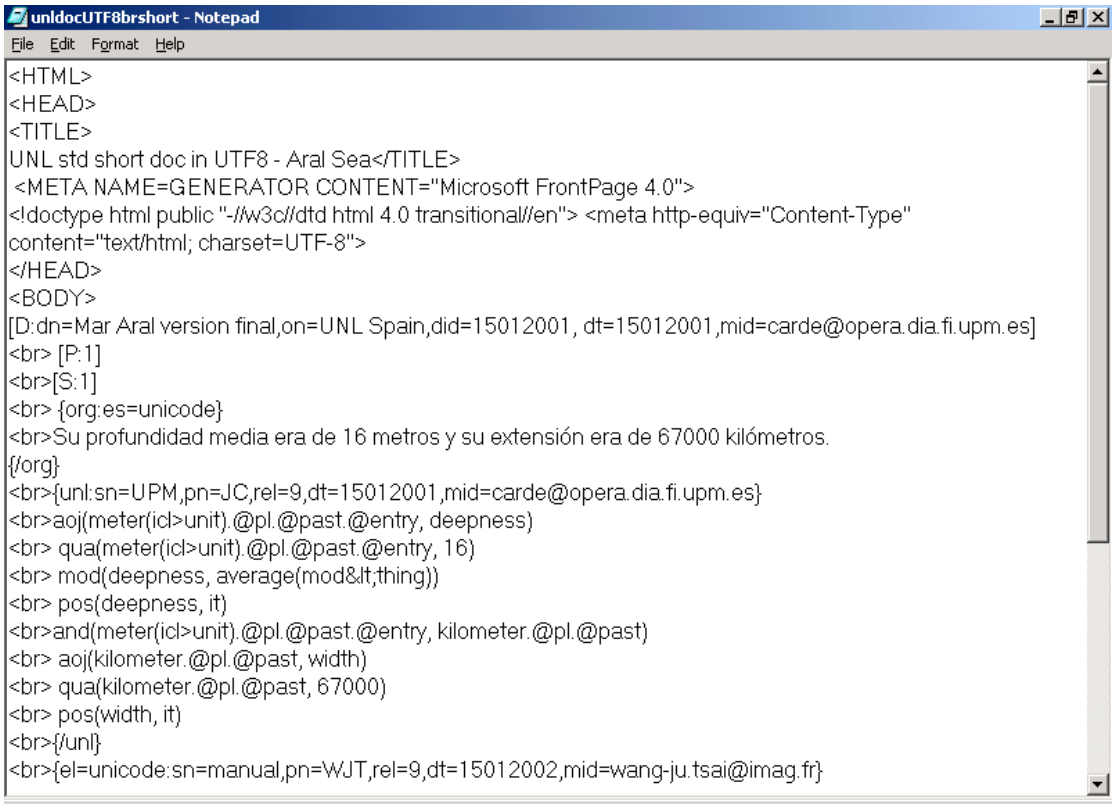


Fig. B-27 Un document UNL-html.2 sous Notepad

Voici un document UNL-html.2 visualisé par un navigateur (Internet Explorer).

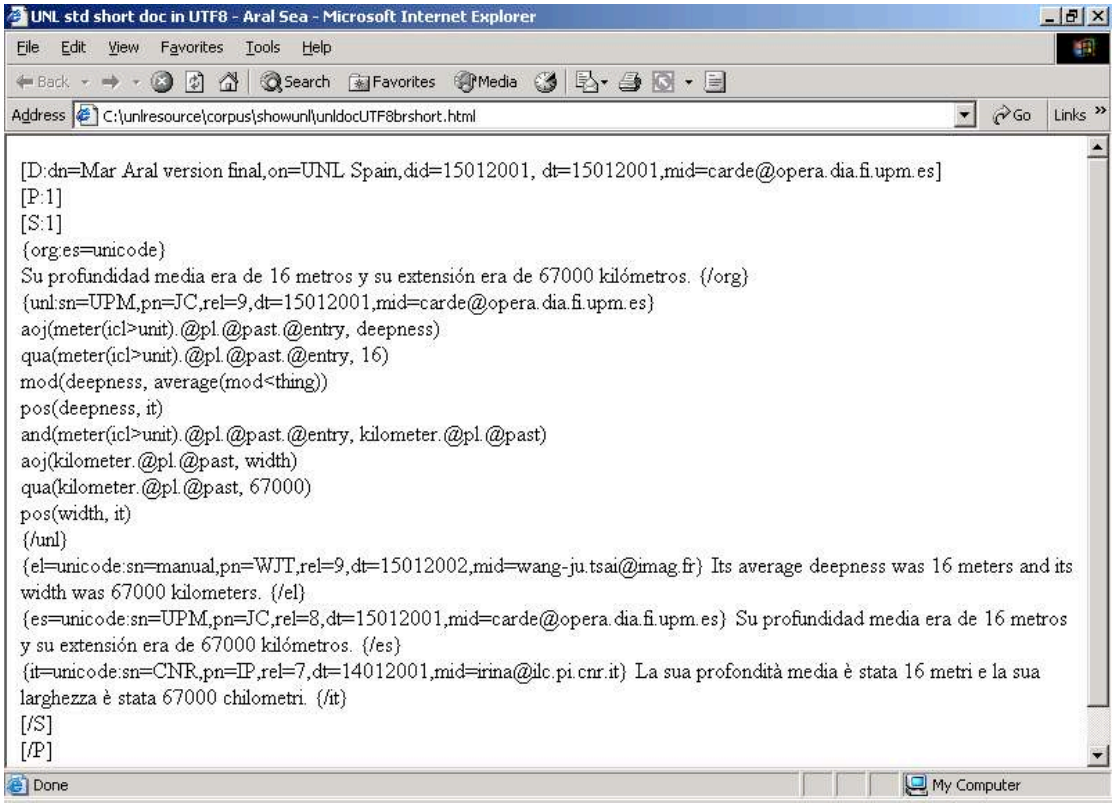


Fig. B-28 Un document UNL-html.2 sous Internet Explorer

La différence entre un document UNL-html.1 et un document UNL-html.2 est simplement qu'il y a des attributs d'encodage et qu'on remplace quelques caractères pour des entités.

Il faut surtout faire attention à échapper le caractère «plus petit que (<) » qui apparaît très souvent dans les restrictions des UW «(mod<thing) ». Si on ne le fait pas, il sera interprété comme une balise ouvrante par le navigateur et l'affichage ne sera pas correct : le texte entre ce caractère et la prochain balise fermante disparaîtra.

Deuxièmement, le résultat d'une déconversion peut contenir des parties du graphe UNL ou des mots entourés par <> (souvent cela veut dire que la déconversion a échoué ou qu'il y a des mots inconnus). Si le résultat est renvoyé par une CGI (Common Gateway Interface), le caractère « plus petit que (<) » sera interprété comme une balise ouvrante, et donc il faut aussi échapper ce caractère.

Les corpus UNL que nous avons étudiés (voir la partie C) sont tous de la forme UNL-html.1, sans échappement de caractères, parce qu'ils viennent directement des corpus qu'on traite ou échange parmi la communauté UNL.

Pour pouvoir utiliser tous les outils développés autour d'UNL, la plupart disponibles gratuitement, nous avons proposé de XML-iser le format de document UNL-html. Nous reviendrons sur ce point dans la partie D.

2.4.2 Visualiser un UNL document sur le web

Il y a deux manières de visualiser un document UNL sur un client web selon le format de ce document UNL. Nous en présentons une ici, et l'autre dans la section D.2.1.6.2.

2.4.2.1 UNL Viewer - pour voir un document UNL-html.1

Développé par le centre UNL en 2000, ce programme ne fonctionne que sur Windows. Il permet à l'utilisateur d'ouvrir un document UNL-html.1 local ou sur le web, et de le visualiser dans la langue de son choix. Le document est alors affiché dans une fenêtre du navigateur utilisé (IE ou Netscape). Ce Viewer a été écrit en Java et sa distribution est limitée à la société UNL.

Voici une figure montrant la structure de ce visualiseur « UNL Viewer ».

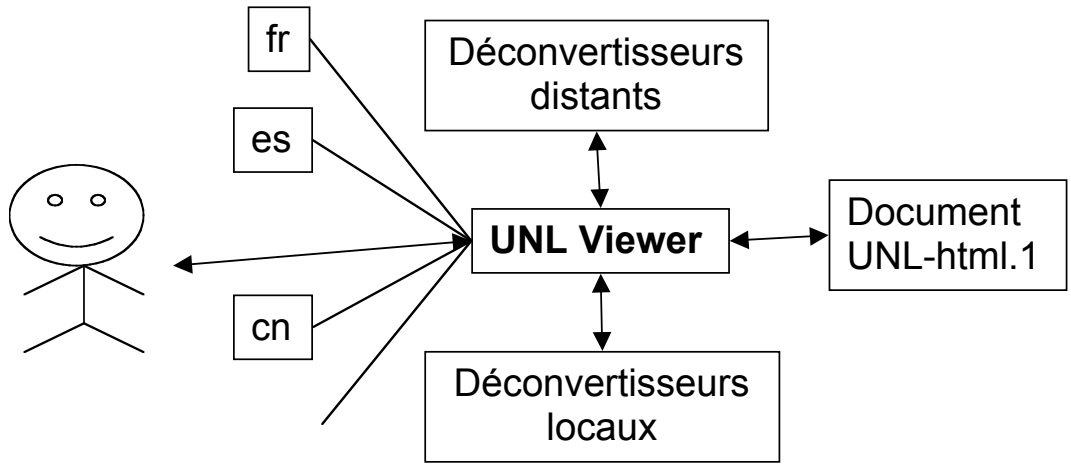


Fig. B-29 Structure du visualiseur « UNL Viewer »

On comprend ici pourquoi le centre UNL n'a jamais explicité le format « réellement html » que nous avons appelé UNL-html.2 : dans sa conception, un document UNL-html.1 n'est jamais directement traité par un navigateur, il est d'abord transformé en un document html classique, produit pour la visualisation dans telle ou telle langue.

Voici l'interface de UNL Viewer :

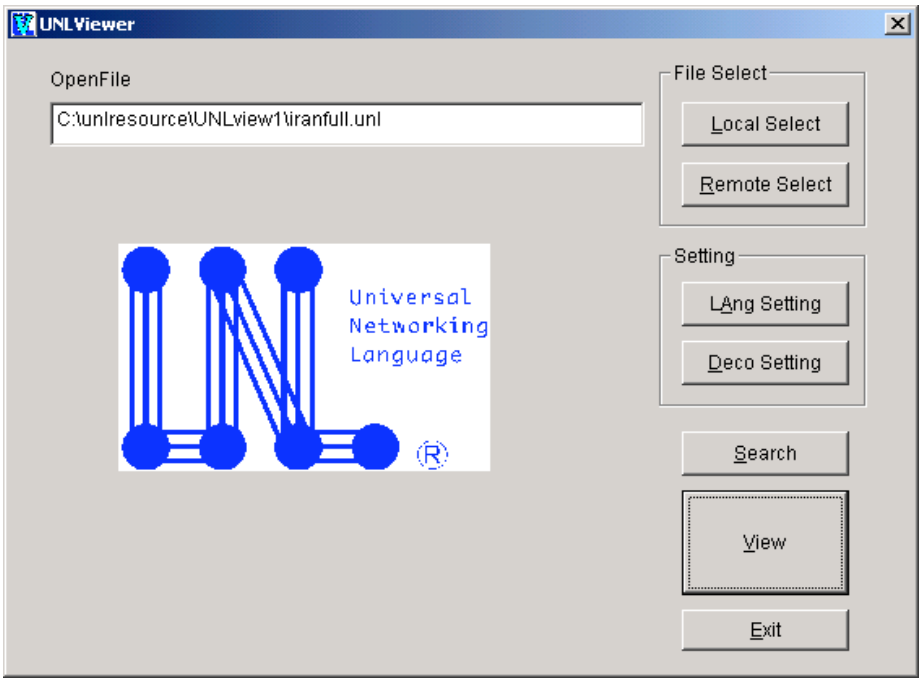


Fig. B-30 Interface du visualiseur « UNL Viewer »

Avant la visualisation, l'utilisateur doit choisir la langue et donner la configuration des déconvertisseurs (locaux ou sur le web).

L'utilisateur peut choisir la langue et le navigateur en cliquant sur « LAng Setting ». Voici l'interface :

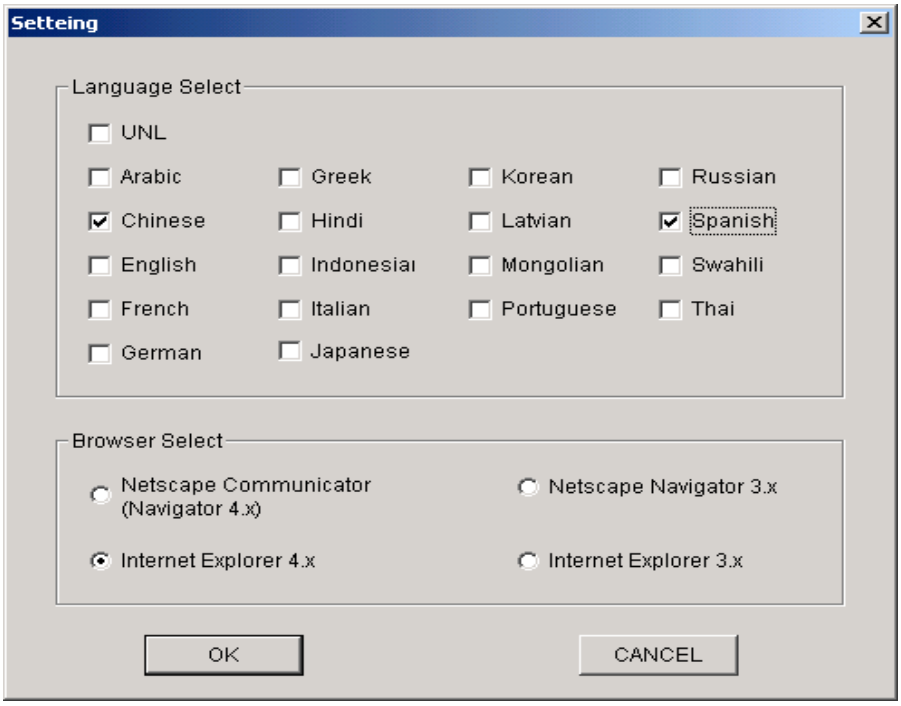


Fig. B-31 Configuration du visualiseur « UNL Viewer »

En cliquant sur « Deco Setting », on paramètre chaque déconvertisseur. Voici une figure où l'utilisateur donne les paramètres du déconvertisseur français.

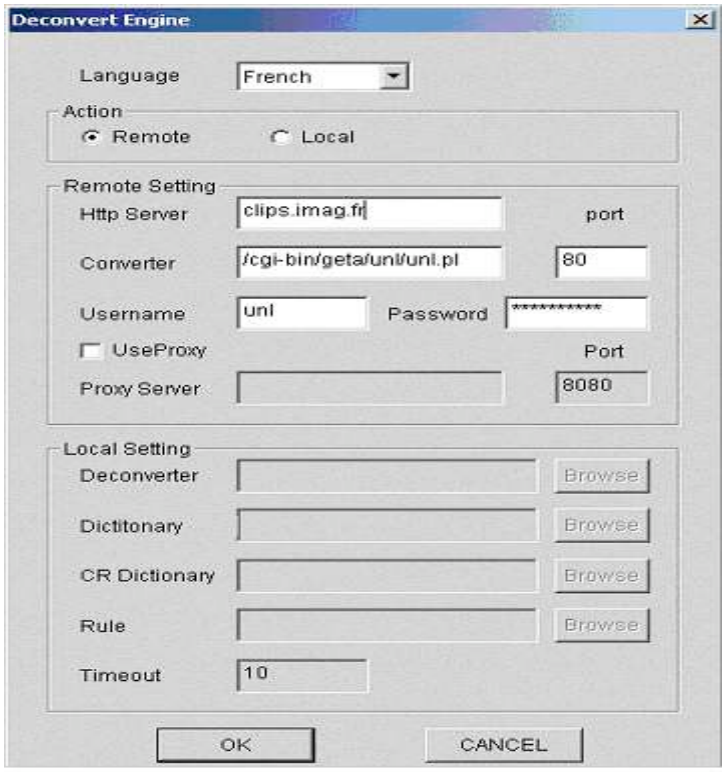


Fig. B-32 Configuration du déconvertisseur français

Voici une figure montrant la visualisation en chinois du document UNL-html de la Fig. B-18.

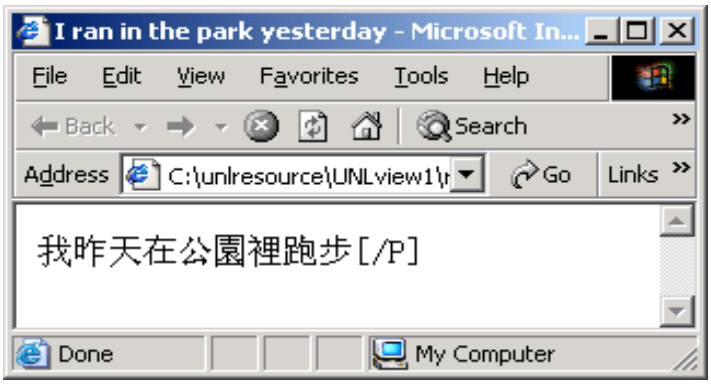


Fig. B-33 Visualisation en chinois sous « UNL Viewer »

3. Conception générale d'un système de coédition fondé sur UNL

3.1 Scénarios

Pour concevoir un environnement de « coédition », il est utile de considérer un scénario type.

3.1.1 Étape 1 : lecture normale

La première étape de notre scénario reprend le scénario initial du projet UNL : un internaute accède à une page web associée à un document UNL, il demande de la lire dans sa langue, et l'obtient. Cela est possible à partir du format UNL-html (avec l'outil « UNL Viewer » du centre UNL) et à partir du format UNL-xml (avec notre visualisateur XSLT).

Il suffit à l'utilisateur de spécifier le document UNL qu'il veut lire.

3.1.2 Étape 2 : un passage manque

Notre environnement devrait permettre à l'utilisateur de spécifier sa préférence de langue de lecture. L'utilisateur choisira donc une ou plusieurs langues dans un certain ordre. Supposons que l'utilisateur a choisi l'ordre de préférence : français, anglais et allemand. Ainsi, le document UNL sera visualisé d'abord en français, et, si la version française n'existe pas, la version anglaise lui sera présentée.

Si un passage manque dans sa langue, c'est que la déconversion n'a pas pu être effectuée (soit il n'y avait pas de graphe UNL soit le déconvertisseur était indisponible). On veut que le lecteur puisse alors très simplement sélectionner le passage en question et « contribuer » sa traduction, s'il est compétent et a le temps.

3.1.3 Étape 3 : lecture « multilingue »

L'utilisateur peut passer au mode de lecture multilingue, s'il connaît plusieurs langues : il peut demander à visualiser plusieurs langues dans une même fenêtre (par exemple, en colonnes parallèles). Cela est tout à fait possible avec les navigateurs commerciaux (on crée un tableau, éventuellement à colonnes réajustables).

3.1.4 Étape 4 : postédition sans coédition

Quand l'utilisateur voit un passage trop mauvais, et sent qu'il peut le corriger, il choisit d'abord ce passage, et entre en mode édition/coédition en cliquant sur un bouton.

Au cas où il n'y a pas de graphe UNL ou de déconvertisseur, une interface de THAM (Traduction Humaine Aidée par la Machine) permettra à l'utilisateur de remplir le passage à la main. Cela est un cas possible mais moins intéressant.

3.1.5 Étape 5 : postédition avec coédition

Si le graphe UNL existe et si le déconvertisseur vers la langue de lecture est disponible, le système proposera à l'utilisateur d'utiliser une interface adaptée à la coédition. Il cliquera sur les parties du passage qu'il veut corriger, et le système lui proposera des choix d'annotation, par exemple, s'il dit « le cheval court dans la plaine », il pourra annoter « le » ou « cheval » ou « court » par « PLURIEL », et « court » par « FUTUR ». Ces annotations seront traduites en modifications sur le graphe UNL, grâce à une correspondance texte-graphe calculable avec peu de ressources (des dictionnaires UNL-langue L et un lemmatiseur de L). L'utilisateur demandera ensuite une nouvelle déconversion de ce passage. Si ça s'améliore, il pourra demander de nouvelles déconversions vers les autres langues dans son profil. L'interface lui permettra de comparer les résultats de coédition dans les langues de son choix.

3.1.6 Étape 6 : postédition avec coédition plus visualisation du graphe UNL

Si l'utilisateur s'intéresse à comprendre mieux la « boîte noire » cachée derrière la coédition, il peut demander au système de visualiser le graphe UNL, les correspondances établies, et le treillis de l'AMS.

3.1.7 Étape 7 : postédition avec coédition plus correction du graphe UNL

Si l'utilisateur est un expert d'UNL ou si le graphe est vraiment trop mauvais et ne s'améliore pas par de simples corrections locales, il est possible que l'utilisateur édite directement le graphe UNL et ensuite demande les nouvelles déconversions.

3.1.8 Étape 8 : retour au contexte de lecture

Quand l'utilisateur est satisfait par les modifications qu'il a faites, le système ajoute dans le document UNL maître ces modifications (versionnage avec monotonie) et l'utilisateur retourne au contexte de lecture.

3.2 Structure du système de coédition utilisant UNL

Nous spécifions trois modes dans cet environnement de coédition : lecture, coédition normale, coédition avancée.

3.2.1 Le mode de lecture

L'utilisateur peut choisir une ou plusieurs langues pour visualiser un document. Si la version de la langue qu'il demande n'existe pas, le système enverra le graphe UNL au déconvertisseur correspondant et récupérera le résultat.

3.2.2 Le mode d'édition normale (pour non-spécialistes)

L'interface permet aux utilisateurs de cliquer sur les mots qu'ils veulent changer. La modification n'est pas libre, elle est faite par un menu qui propose des choix possibles liés à ce mot. Il faut aussi garder un champ pour que l'utilisateur puisse entrer le texte manuellement quand le graphe UNL ou le déconvertisseur de la langue en question n'existe pas. La procédure de liaison du texte et du graphe peut être trop compliquée et il vaut mieux la cacher dans la boîte noire du système.

3.2.3 Le mode d'édition avancée pour les experts

Il nous semble nécessaire d'avoir un mode d'édition pour les spécialistes. Dans ce mode, les spécialistes peuvent voir et manipuler les graphes UNL en forme textuelle et graphique, ainsi que toutes les traces de la coédition, y compris les correspondances.

L'intérêt d'ouvrir cette boîte noire aux utilisateurs est multiple:

> On peut remplacer les nœuds par les mots de la langue de l'utilisateur comme le fait l'équipe espagnole pour faciliter la compréhension du graphe.
>
> Parfois, il est plus facile de manipuler la forme graphique directement : il est donc important qu'elle soit facile à comprendre. Pour cela, on pourra la « localiser » (la présenter dans la langue de lecteur).
>
> Il existe des erreurs qu'on ne peut corriger que sur le graphe UNL.
>
> Les progrès dans le domaine de l'IIHM (ingénierie de l'interaction homme-machine) nous donnent l'espoir qu'un jour une interface facile à comprendre et à manipuler sera produite.
>
> Grâce aux jeux vidéos, les jeunes manipulent des interfaces complexes beaucoup mieux aujourd'hui, et donc on peut espérer qu'une interface plus compliquée que du texte simple sera acceptée.

3.2.4 Erreurs corrigibles et non corrigibles

Il est important de noter qu'on ne peut pas espérer pouvoir corriger toutes les erreurs par coédition, même s'il y a un graphe UNL. En effet, ce graphe peut être faux par rapport au texte d'entrée, et le déconvertisseur lui-même peut aussi être « trop faux ». Enfin, on a vu qu'il y a des finesses « inexprimables » en UNL, comme d'ailleurs dans tout interlingua.

Nous pourrons corriger le graphe par coédition, mais nous ne pourrons pas corriger les erreurs provenant du déconvertisseur. Nous traitons l'enconvertisseur et le déconvertisseur comme des « boîtes noires ».

Les erreurs sur les graphes peuvent être encore divisées en deux groupes : erreurs syntaxiques et erreurs sémantiques.

Erreur syntaxique - Les graphes ne respectant pas les spécifications UNL sont dits syntaxiquement erronés. Par exemple, manque de parenthèse fermante, graphe non-connecté, faute d'orthographe dans l'écriture des relations ou attributs, UW non-consistante, numérotation d'identité non-consistant, UW absente de la KB, etc. Ce genre d'erreur doit être signalé par le vérificateur UNL ou les déconvertisseurs avant la déconversion et donc ne devrait pas apparaître. Malheureusement, en pratique, ce genre d'erreur existe encore dans les corpus qui circulent.

Erreur sémantique – il s'agit de graphes sous-spécifiés, de graphes qui ne représentent pas exactement le texte original (dans le cas où le texte original est présent), ou encore d'emploi erroné des relations ou des attributs, et d'erreurs lexicales peut-être cachées en langue source mais apparentes en langue cible, etc.

La procédure de coédition traite donc ce dernier type d'erreur.

3.3 Architecture interne à quatre niveaux

Pour réaliser la coédition, il faut établir les liaisons entre le graphe UNL et le texte, de façon à pouvoir manipuler le graphe UNL à travers le texte. Personne n'a encore étudié la modélisation de la correspondance graphe-texte, et cela paraît difficile à faire directement. Nous proposons donc de décomposer une telle correspondance en quatre niveaux, et d'établir les correspondances entre deux niveaux voisins, puis de faire le produit de ces relations. Les niveaux retenus sont le graphe UNL, l'arbre UNL, le treillis LMS (lexico-morphosyntaxique), et le texte.

3.3.1 Graphe-UNL

C'est un graphe UNL conforme aux spécifications. Il peut avoir été produit à la main ou automatiquement par un enconvertisseur, ou semi-automatiquement.

3.3.2 Texte

C'est un énoncé en langue naturelle qui est censé correspondre au graphe UNL en question. Normalement, ce texte vient de la déconversion du graphe UNL. Si on est dans l'environnement de coédition, c'est que le lecteur a détecté des erreurs. Il s'agit de les corriger en 2 temps : corriger le graphe, puis déconvertir à nouveau.

3.3.3 Treillis-LMS

Le texte ne contient que des formes, difficiles à mettre en correspondance avec des UW. Nous cherchons donc un niveau intermédiaire contenant des lemmes, faciles à mettre en correspondance avec des UW via des dictionnaires langue L-anglais (ou langue L-UNL).

Le seul niveau d'analyse contenant ces informations et obtenable gratuitement et pour la majorité des langues informatisées est celui de l'analyse morpho-syntaxique.

Tous les segmenteurs, lemmatiseurs et analyseurs morpho-syntaxiques produisent des résultats qui sont des treillis ou qu'on peut très facilement transformer en treillis.

C'est pourquoi nous considérons un niveau dit « treillis LMS » (lexico-morpho-syntaxique).

3.3.4 Arbre-UNL

Un « arbre-UNL » est une structure d'arbre correspondant directement et bijectivement à un graphe UNL [Blanc 00, 01][Sérasset 99]. La procédure de transformation est relativement simple. Nous l'expliquerons dans la section C.3.3.2.

Il est possible que les correspondances entre le graphe UNL et le treillis puissent s'exprimer par un ensemble de liaisons comme dans [Planas 98]. Mais nous préférons passer par « l'arbre UNL », qui est facile à obtenir, et permet plus facilement de montrer les correspondances. Du point de vue théorique, cela nous permettra aussi d'appliquer les études existantes sue les correspondances entre arbres et chaînes pour calculer les correspondances désirées.

3.4 Résumé de la démarche

Il est facile d'exprimer les correspondances entre éléments d'un texte et nœuds d'un treillis LMS. Mais il n'existe pas de modèle de mise en correspondance d'un graphe UNL (hypergraphe avec sous-graphes « repliés ») avec un treillis LMS (lexico-morpho-syntaxique).

Nous nous appuierons donc sur le fait qu'il existe des modèles plus ou moins élaborés de correspondances entre chaînes et arbres (concrets et abstraits). Nous nous référons en particulier au modèle chaîne-arbre de [Boitet & Zaharin 88] dérivé de celui de la STCG (String-Tree Correspondence Grammars [Zaharin 86c, 87]), dans lequel les correspondances sont encodées par deux fonctions, SNODE et STREE (voir Partie C section 1.3 pour les définitions formelles) appliquant les nœuds de l'arbre dans les séquences (connexes ou non) d'items lexicaux de la chaîne, ce qui se généralise immédiatement au cas d'un treillis LMS.

A ce point, on remarque qu'il existe des algorithmes standard et réversibles de transformation d'un graphe-UNL en un « arbre-UNL » [Serasset 99] et réciproquement [Blanc 00, 01]. En un sens, cette dernière correspondance n'est pas statique comme les deux autres, mais elle nous suffira.

C. Étude des correspondances UNL-texte

Introduction

Nous avons choisi UNL comme pivot interlingue et présenté des critères pour concevoir un système de coédition fondé sur UNL dans la partie précédente. Nous abordons maintenant la question de savoir comment nous pouvons construire des liens (correspondances) entre une structure interne (un graphe UNL) et un texte, a posteriori.

Comme dit précédemment, nous divisons cette correspondance en 3 correspondances :

correspondance graphe UNL et arbre UNL.

correspondance arbre UNL et treillis LMS (lexico-morpho-syntaxique).

correspondance treillis LMS et texte.

Nous détaillerons surtout la seconde, définie à partir de la théorie des grammaires statiques, et un peu moins la première (nous nous bornerons à montrer une procédure la réalisant). Quant à la troisième, elle est presque immédiate.

Nous étudions ensuite les corpus UNL dont nous disposons. Cette étude nous conduit à définir une hiérarchie de graphe et de texte, et à essayer d'identifier des correspondances entre différents niveaux de ces deux hiérarchies.

Enfin, nous proposons un algorithme pour construire les liens entre le graphe UNL et le texte.

1. Modélisations de correspondances entre structures

Nous commençons par une revue des études antérieurs sur la modélisation des correspondances entre chaînes et arbres. Depuis 1983, 5 modèles principaux ont été proposés.

1.1 Grammaire statique (Chappuy 1983, Vauquois et Chappuy 1985)

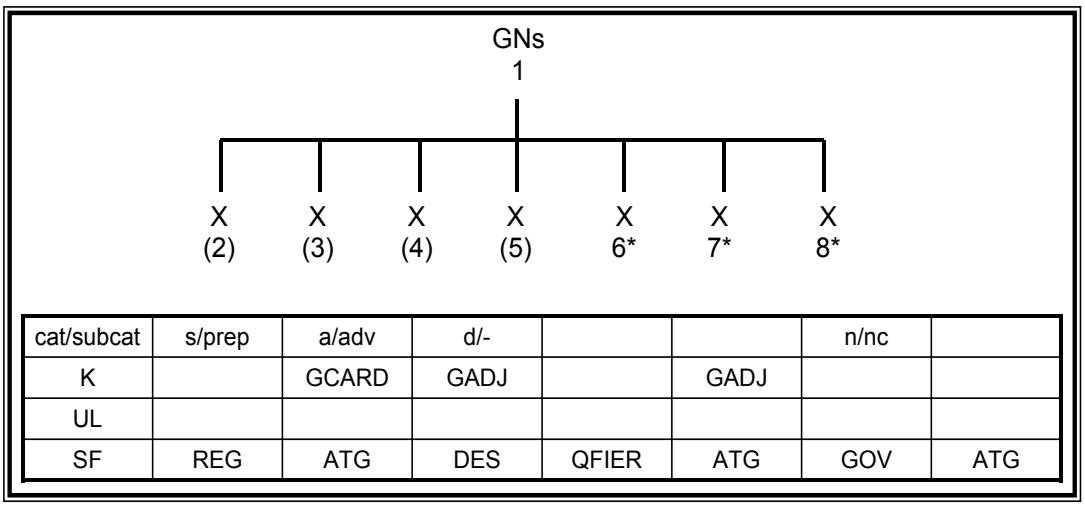
[Vauquois 85a] [Zajac 86] [Zaharin 86a] [Grasson 96]

Dès1980, Vauquois commençait à essayer de représenter les correspondances entre un énoncé et la m-structure (multi-niveau) correspondante par des diagrammes simples [Boitet 90a]. En effet, à cette époque-là, il n'existait pas de formalisme satisfaisant pour cela. Après une vingtaine d'années de développement en TA, et avec l'expérience de la programmation dynamique en ROBRA depuis 5 ans, Vauquois pensait qu'il était temps de formaliser la description des correspondances entre énoncés et arbres abstraits, pour arriver à construire une méthode de « génie linguiciel » reposant sur un niveau de spécification formelle. Il a travaillé sur cette idée avec Chappuy, qui a formalisé le résultat de ces recherches introduisant le tout premier formalisme des Grammaires Statiques dans sa thèse [Chappuy 83].

Une grammaire statique se compose de « planches statiques » : une planche établit la correspondance entre un arbre et une famille de chaînes, ainsi que les contraintes sur cette structure et cette famille.

Une planche se décompose en trois zones : la zone 1 définit la correspondance entre la structure et la famille de chaînes ; la zone 2 définit les contraintes sur les chaînes ; la zone 3 définit la mise en place de la décoration portée sur la structure. En plus, il y a une en-tête de planche qui spécifie son nom, des informations de gestion et des commentaires, et aussi les planches référées par cette planche.

Voici un exemple de zone 1. La structure est représentée par un schéma d'arbre décoré et la famille de chaînes est représentée par une liste de nœuds décorés. La décoration des nœuds des arbres et des chaînes contient des informations des différents niveaux linguistiques intervenant dans un m-structure (niveaux interlingues des prédicats et arguments logiques, des relations et attributs sémantiques, et de l'actualisation abstraite, ainsi que niveaux plus « surfaciques » comme des catégories et fonctions syntaxiques et syntagmatiques) du GETA. Les chiffres représentent les différents éléments dans la chaîne.



cat/subcat	s/prep	a/adv	d/-			n/nc	
K		GCARD	GADJ		GADJ		
UL							
SF	REG	ATG	DES	QFIER	ATG	GOV	ATG

Fig. C-1 Zone 1 de Grammaire Statique

Si la présence d'un élément est facultative, cet élément est mis entre parenthèses. Le signe * représente l'itération libre de l'élément ; le signe + signifie que l'élément doit être itéré au moins une fois ; le signe x signifie que l'élément apparaît dans la chaîne ; et le signe o signifie que l'élément apparaît dans la structure mais pas dans la chaîne.

La zone 2 définit les contraintes sur la chaîne, c'est-à-dire l'absence ou la présence de mots selon l'absence ou la présence d'autres mots. Des variables et opérateurs sont aussi utilisés. Voici un exemple de zone 2 d'un groupe nominal simple (Fig. C-2 ci-dessous) :

3 - IMP - 4 - OU - 5
si 3 est présent dans la chaîne alors 4 ou 5 est présent
$AGR.GNR.NBR(4,5,6,7,8)
accord en genre et en nombre entre les sommets 4,5,6,7,8.

Dans l'exemple, « les deux jolis chats gris » : les nœuds 5,6,7,8 ont mêmes attributs en genre et en nombre : masculin et pluriel.

Fig. C-2 Zone 2 de Grammaire Statique

La zone 3 représente la mise en place de la décoration portée par la structure. Cette zone peut encore être divisée en deux parties : la première partie spécifie la correspondance directe de la décoration et de la structure. La deuxième partie contient les contraintes sur les sommets.

Voici la première partie de la zone 3, qui exprime ici que le genre et le nombre du nœud 1 sont égaux à ceux des nœuds 4 et 5 et 6 et 7 et 8.

Fig. C-3 Première partie d'une zone 3 de Grammaire Statique

La deuxième partie de la zone 3 se compose d'une ou de plusieurs contraintes ordonnées. L'ordre d'évaluation est de haut en bas. L'exemple suivant spécifie deux contraintes sur le nœud 4 :

si le nœud 4 (de la chaîne) a pour sous-catégorie « déicteur article indéfini », alors la détermination du nœud 1 (de l'arbre) est indéfinie.

si le nœud 4 a pour sous-catégorie « déicteur article défini », démonstratif ou possessif, alors la détermination du nœud 1 est définie.

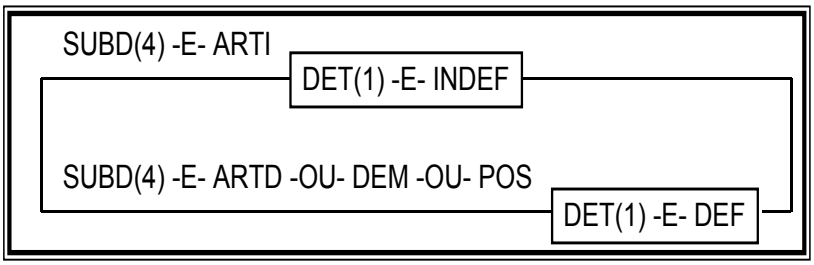


Fig. C-4 Deuxième partie d'une zone 3 de Grammaire Statique

Dans l'en-tête de planche, on peut spécifier le nom, la langue, le type et les autres planches référées.

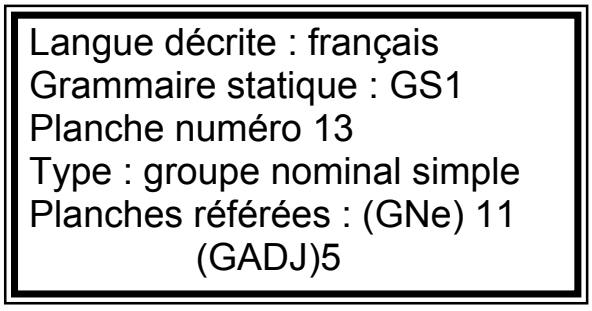


Fig. C-5 En-tête d'une planche

Nous donnons un exemple complet d'une planche de GS dans l'Annexe F.

Chappuy a défini une hiérarchie entre les planches, qui permet de mettre des contraintes systématiques (et implicites) sur l'emploi des références. Il y a trois groupes de planches : planches élémentaires, simples et complexes. Cette hiérarchie définit l'ordre logique pour construire les planches. Par exemple, une planche simple ne peut pas référer à une planche complexe.

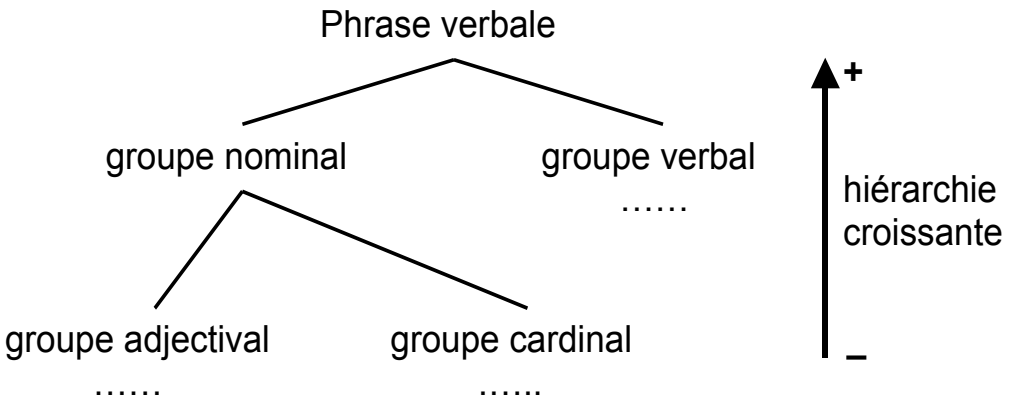


Fig. C-6 Hiérarchie des planches

La théorie de la GS est le résultat de l'expérience accumulée en réalisant de gros linguiciels par programmation dynamique (ou procédurale), principalement en ROBRA. C'est pourquoi le premier formalisme des GS est (syntaxiquement) très influencé par celui de ROBRA [Boitet 78].

Si on utilise une GS comme niveau de spécification, on sépare bien la spécification de l'implémentation, par exemple, d'un analyseur. Pour construire un analyseur efficace, construisant une « meilleure » solution de façon heuristique, il faut en effet définir aussi une stratégie, et combiner les deux. Malheureusement, personne n'est encore arrivé à décrire une telle stratégie de façon complète et autonome. Si on le pouvait, on aurait la figure suivante.

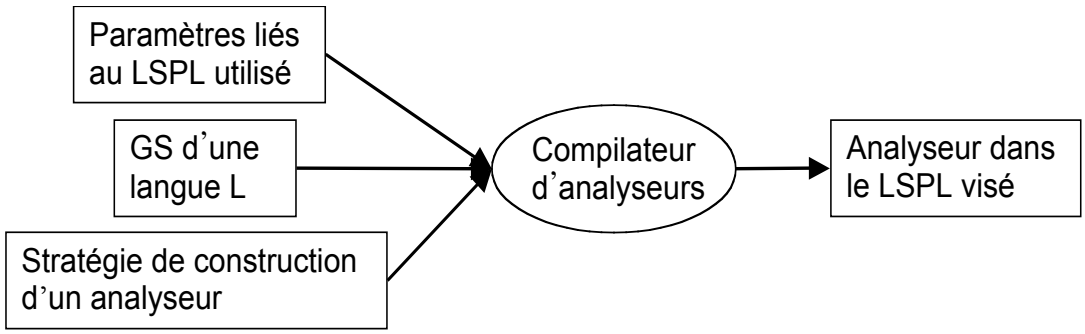


Fig. C-7 Utilisation idéale d'une GS pour construire des analyseurs

En fait, on peut arriver à créer des squelettes d'analyseurs (ou de générateurs) qu'il faut ensuite compléter et mettre au point à la main. C'est ce qu'a fait Tang Enya Kong [Tang 94] dans sa thèse.

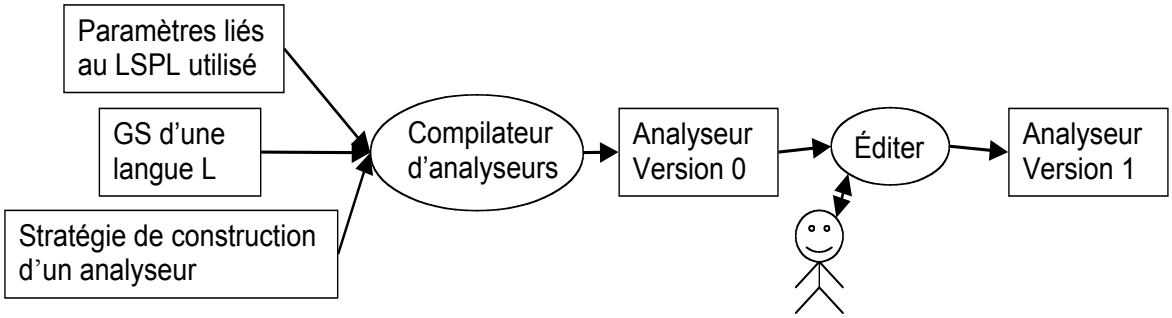


Fig. C-8 Mise au point d'un analyseur à la main

Les travaux subséquents sur les GS sont nombreux : comme

TCG (Tree Correspondence Grammar) de Zaharin [Zaharin 86a],

CSCG (Grammaire de Spécification de Correspondances Structurales) de Zajac [Zajac 86a],

GCI (Grammaires Correspondancielles d'Identification) de Lepage [Lepage 89],

STCG (String-Tree Correspondence Grammar) de Zaharin et Tang [Tang 94].

Ces différentes études avaient toutes pour objectif d'élaborer un formalisme plus simple et plus rigoureux, permettant d'obtenir une sémantique formelle très précise, et donc de progresser vers plus d'automaticité dans la construction automatique d'analyseurs et de générateurs.

1.2 String-Tree Correspondence Grammars «STCG» (Zaharin, 1987)

[Zaharin 86a] [Grasson 96]

La STCG a été anciennement nommée « Tree Correspondence Grammar (TCG) » dans [Zaharin 86a] comme un raffinement de la Grammaire Statique.

Comme la GS, c'est aussi un système déclaratif qui permet aux linguistes de spécifier la correspondance entre un ensemble d'arbres et un ensemble de chaînes, et les sous-correspondances entre sous-arbres et sous-chaînes. Le choix de la classe des structures arborescentes (à constituants, lexicalisées, dépendancielles..) est assez libre, donc la STCG est assez indépendante des théories grammaticales.

Pour spécifier les correspondances plus fines (par exemple correspondance croisée ou discontinue), la STCG a été complétée par un dispositif d'annotation dit « SSTC » (Structured String-Tree Correspondence) [Boitet 88], que nous présenterons plus loin.

Comme la GS, la STCG se compose de planches, chaque planche définissant une règle de correspondance. Une planche se décompose en deux parties : la partie gauche spécifie la correspondance principale ; la partie droite spécifie explicitement les sous-correspondances (correspondances entre les sous-arbres et les sous-chaînes).

Voici une règle ou planche d'une STCG :

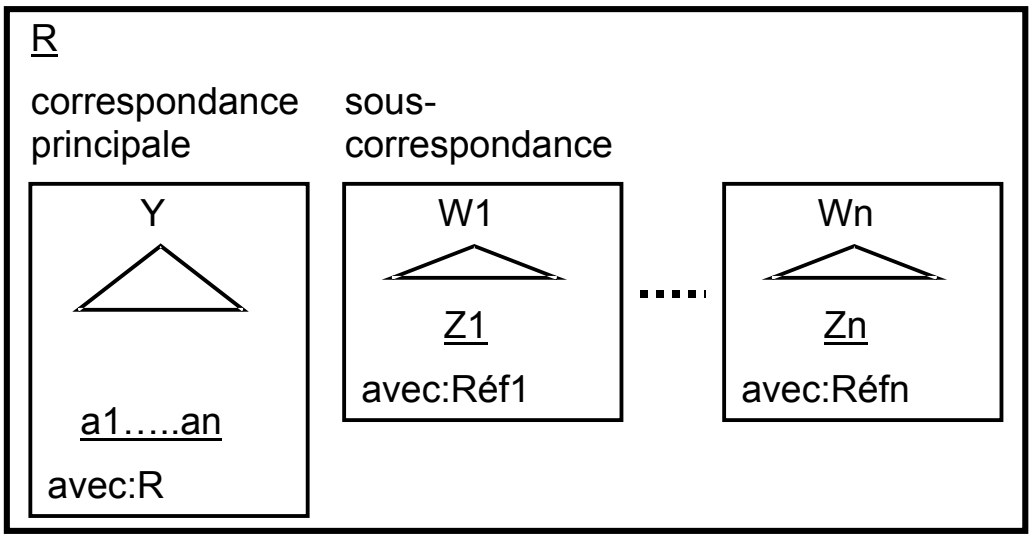


Fig. C-9 Une planche de STCG

Chaque planche a un nom et spécifie une correspondance entre une famille d'arbres (forêt) et une famille de chaînes.

La syntaxe d'une règle de STCG est suivante :

(le contenu entre parenthèses est facultatif et le signe * signifie l'itération.)

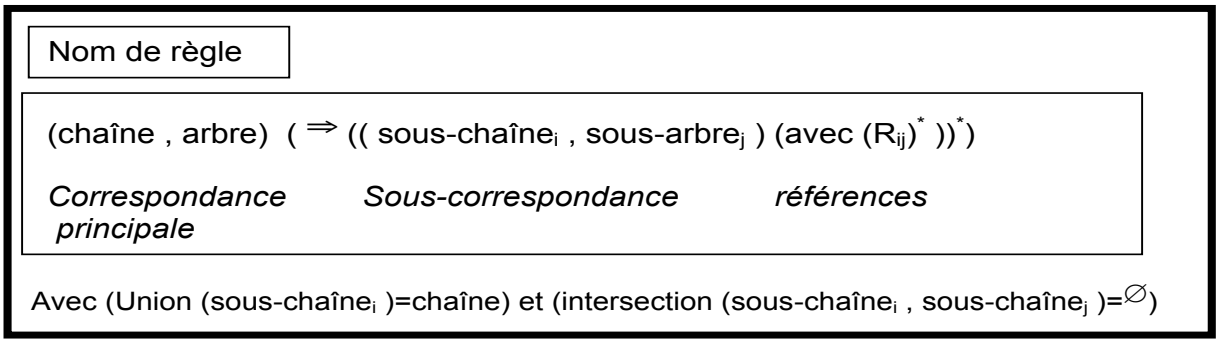


Fig. C-10 Syntaxe d'une règle de STCG

Voici un exemple de STCG avec trois règles qui spécifie le groupe nominal :

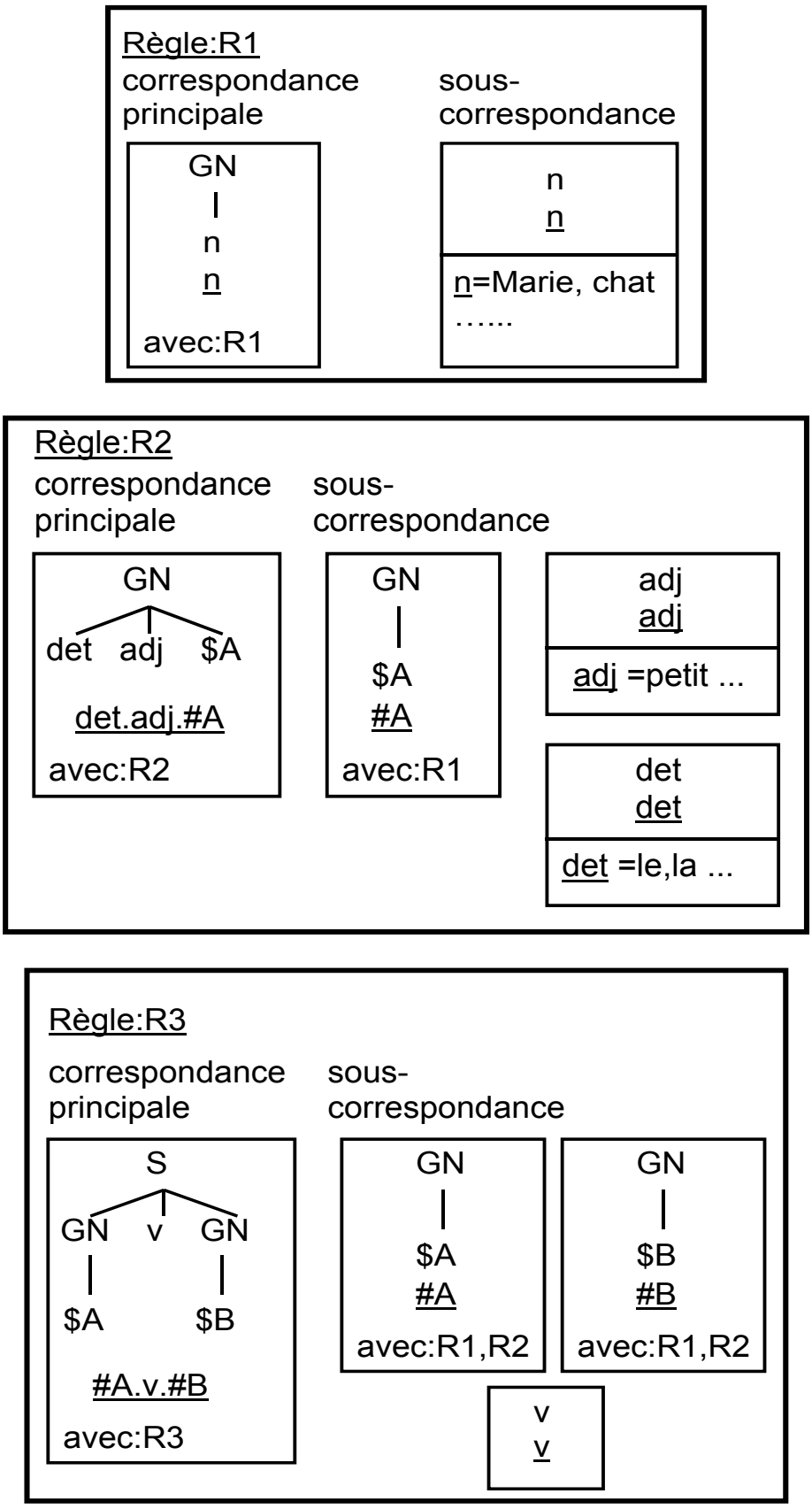


Fig. C-11 3 planches de STCG pour le groupe nominal

Dans cet exemple, la planche R3 réfère à R1 et R2 dans ses sous-correspondances.

Les travaux ultérieurs sur les STCG ont été effectués dans l'équipe de l'UTMK, USM (Universiti Sains Malaysia), dirigée par Zaharin. Tang a fait sa thèse [Tang 94] sur l'application des STCG à l'analyse des langues naturelles. Un éditeur de GS a été réalisé par Y. Lepage (SaGE, Static Grammar Graphic Editor) et a ensuite été modifié pour accepter le formalisme des STCG. Enfin, Tang a produit un générateur de systèmes transformationnels de génération en ROBRA à partir d'une STCG.

1.3 Structured String-Tree Correspondences «SSTC» (Boitet & Zaharin, 1988)

Selon la théorie Sens-Texte de Mel'_uk (_____), la langue naturelle est une correspondance entre des niveaux différents de représentation, le sens et le texte étant les deux niveaux les plus éloignés [Mel'_uk 65] [Tang 95]. Cependant, la langue naturelle n'est pas simplement une correspondance entre représentations, mais aussi une correspondance entre « sous-correspondances ». Une annotation pour la description de correspondances doit satisfaire ce critère.

Dans [Boitet 88a], Boitet et Zaharin ont proposé une telle annotation pour mieux décrire la correspondance entre la chaîne et la structure abstraite (très souvent un arbre). L'intérêt de la SSTC est sa souplesse et sa capacité de décrire la correspondance entre une chaîne et un arbre abstrait, ainsi que les correspondances entre sous-chaînes (éventuellement discontinues) et sous-arbres, ces correspondances pourrant être non-projectives. Cette caractéristique est très désirable quand on veut décrire des phénomènes linguistiques non-standard. [Al-Adhaileh 98] caractérise les phénomènes non-standard comme suit.

Fusion des nœuds

Soit le graphe suivant qui représente la correspondance entre la phrase anglaise « He picks up the ball » et son arbre prédicat-argument. On a un cas où une particule (pouvant parfois être une préposition – « up the street ») fait partie d'un prédicat au lieu d'introduire un argument ou un circonstant. Et donc « pick » et « up » n'occupent qu'un seul nœud dans l'arbre.

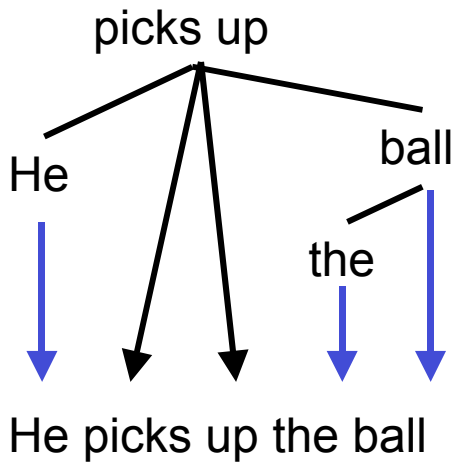


Fig. C-12 Correspondance dans un cas de fusion de deux nœuds

Omission des mots

La figure suivante montre la correspondance entre la phrase anglaise « John eats the apple and Mary the pear » et son arbre prédicat-argument. A cause de l'élision du second « eats », le mot « eat » correspond en fait à deux nœuds dans l'arbre.

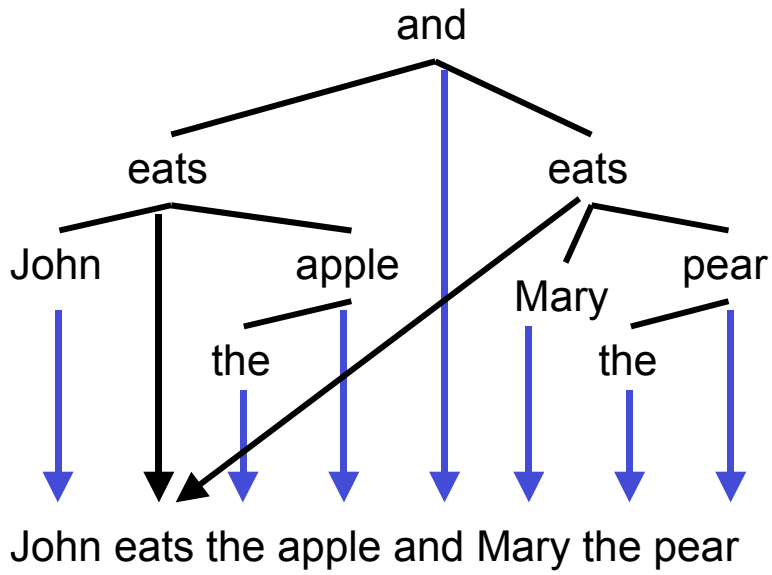


Fig. C-13 Correspondance dans le cas d'une élision

Dépendances croisées

Un cas plus compliqué dans la correspondance chaîne-arbre est que les dépendances se croisent, et pourtant cela n'est pas rare dans la langue naturelle.

C'est le cas où un sous-arbre correspond à une sous-chaîne, mais où les mots de cette sous-chaîne se distribuent dans la chaîne entière.

C'est aussi le cas des phrases de forme ($a^n\ vb^n c^n$ avec n >0). Un exemple en anglais est « John and Mary give Paul and Ann trousers and dresses [respectively] ». La figure suivante montre cette correspondance complexe.

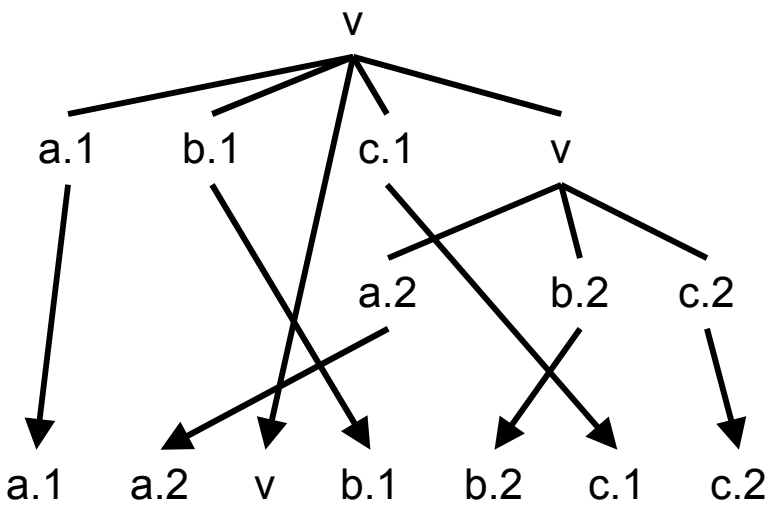


Fig. C-14 Dépendance croisée

Dans d'autres cas, on trouve aussi un mélange de ces correspondances non-projectives. Par exemple, la figure suivante montre que la correspondance entre la phrase « He picks the ball up » et son arbre de dépendance abstrait fait apparaître une fusion de nœuds et une dépendance croisée.

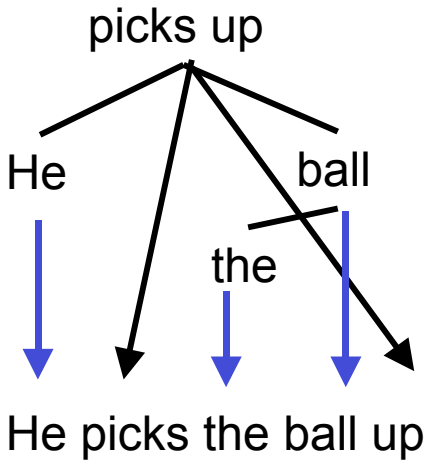


Fig. C-15 Dépendance croisée et fusion des nœuds

Pour résoudre les problèmes que nous venons de voir, il faut avoir une annotation assez souple et puissante pour décrire toutes ces correspondances non-projectives. Cette annotation décrira la correspondance à deux niveaux : nœud – texte et arbre - syntagme (continu ou discontinu). C'est le but de la SSTC.

Nous donnons maintenant la définition formelle de la SSTC [Al-Adhaileh 02a] :

> Une **SSTC** est une structure générale qui associe un arbre arbitraire (sa structure d'interprétation) à une chaîne dans une langue naturelle, et aussi la correspondance entre la chaîne et l'arbre, qui peut être non-projective. Donc une **SSTC** est un triplet (**st, tr, co**), où :
> **st** est une chaîne
> **tr** est un arbre associé à cette chaîne

co est la correspondance entre la chaîne et l'arbre.

La correspondance **co** entre la chaîne et l'arbre se compose de deux correspondances en corrélation :
a) entre les nœuds et sous-chaînes (qui peuvent être discontinues).
b) entre sous-arbres (qui peuvent être incomplets) et sous-chaînes (possiblement discontinues).

La correspondance est enregistrée dans chaque nœud **N** de l'arbre par deux séquences d'intervalles, **SNODE(N)** et **STREE(N)**.

SNODE(N) contient la sous-chaîne (peut-être discontinue) qui correspond à ce nœud **N** de l'arbre.

STREE(N) contient la sous-chaîne (peut-être discontinue) qui correspond au sous-arbre dont la racine est ce nœud **N**.

Voici un exemple de SSTC, qui décrit la correspondance entre la phrase anglaise « John picks up the ball » et son arbre prédicat-argument. Chaque nœud est de la forme « lemme (SNODE/STREE) ». Nous spécifions aussi STREE et SNODE du nœud « ball ».

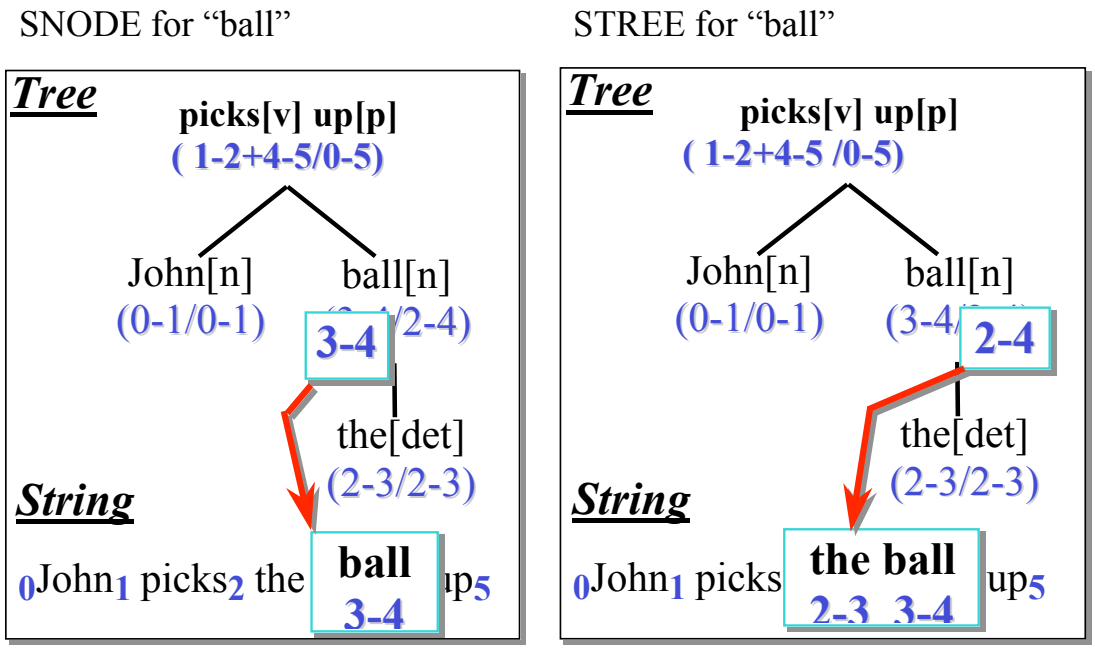


Fig. C-16 Exemple de SSTC pour une correspondance non-standard

Dans la figure suivante, nous montrons qu'une SSTC peut aussi décrire la correspondance entre une chaîne et sa structure syntagmatique. Dans ce cas, la valeur de SNODE sur les nœuds non-terminaux est ∅.

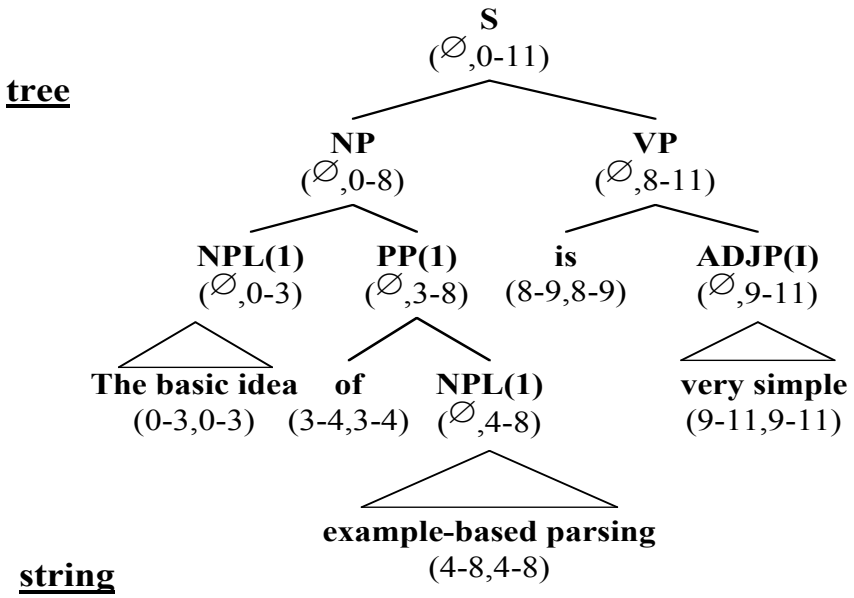


Fig. C-17 SSTC pour un arbre syntagmatique

Zaharin a appliqué l'annotation SSTC (SNODE, STREE) au formalisme des STCG. Ensuite, Tang a développé l'application des STCG, toujours en utilisant l'annotation SSTC, à l'analyse de la langue naturelle [Tang 94]. Plus tard, dans [Al-Adhaileh 99], la théorie des SSTC a été étendue pour décrire la correspondance entre deux couples (texte, arbre) comme une « synchronisation », en fait une correspondance « horizontale », entre les deux SSTC reliant $texte_i$ à $arbre_i$ (i=1,2).

1.4 Synchronous SSTC «S-SSTC» (Tang & Mosleh, 1999)

Une SSTC est une annotation décrivant la correspondance entre une structure de représentation et une chaîne. [Al-Adhaileh 99] a développé plus loin cette idée et a proposé de synchroniser deux correspondances chaîne-arbre afin d'obtenir une correspondance entre deux énoncés de deux langues différentes.

Basée sur une SSTC, une SSTC synchrone (S-SSTC) contient exactement les mêmes correspondances entre chaînes et arbres que cette SSTC. On y ajoute les informations IndexStree et IndexSnode pour enregistrer les correspondances entre les morceaux des deux SSTC.

Comme la SSTC, qui peut décrire des correspondances non-standard entre la structure et le texte, la S-SSTC peut décrire des correspondances non-standard entre deux langues. Nous pouvons citer au moins trois phénomènes non-standard dans la correspondance entre deux énoncés de deux langues : a) la correspondance N_1 (non-injective), b) l'élimination de dépendance, c) l'inversion de dépendance.

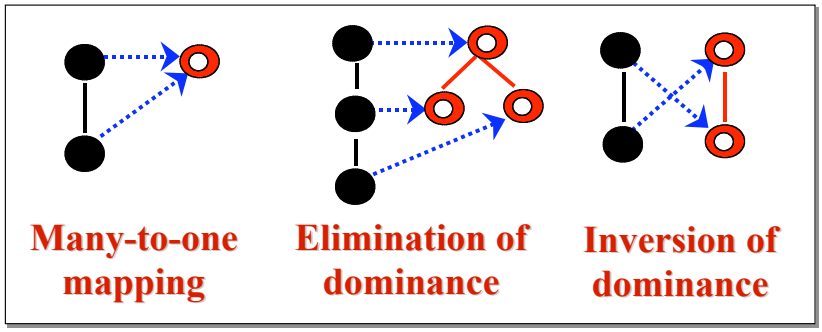


Fig. C-18 Quelques correspondances non-standard entre deux langues

Voici la définition formelle d'une S-SSTC :

- **S** et **T** sont deux triplets de **SSTC**, i.e., 2 triplets de type (**st, tr, co**), à savoir (**st**(S), **tr**(S), **co**(S)) et (**st**(T), **tr**(T), **co**(T)), où **st**(S) est la chaîne de S, **tr**(S) l'arbre de S, et **co**(S) la correspondance SNODE/STREE entre **st**(S) et **tr**(S).
- Une SSTC synchrone S_{syn} est un triplet (S, T, $\varphi_{(S,T)}$), où $\varphi_{(S,T)}$ est un ensemble de liens qui définissent les correspondances synchrones[13] entre les nœuds de **tr**(S) et les nœuds de **tr**(T), aux différents niveaux des deux SSTC.
- Pour chaque unité élémentaire (c'est-à-dire, nœud, sous-arbre, ou sous-arbre partiel) N_S dans la première SSTC S, il existe N_T (une ou plusieurs unité/s élémentaire/s) dans la deuxième SSTC T qui correspond/correspondent à N_S.
- Pour chaque paire (N_S, N_T), telle que N_S correspond à N_T, il existe entre N_S et N_T un lien $l \in \varphi_{(S,T)}$.
- Un lien $l \in \varphi_{(S,T)}$ qui peut être du type l_{sn} ou l_{st} définit les correspondances synchrones entre les nœuds de **tr**(S) et les nœuds de **tr**(T).
- Une correspondance du type l_{sn} spécifie la correspondance entre un nœud N_S et un nœud N_T .
- Une correspondance du type l_{st} spécifie la correspondance entre le sous-arbre de racine N_S et le sous-arbre de racine N_T.
- les correspondances synchrones l_{sn} et l_{st} peuvent lier deux nœuds et deux sous-arbres de façon non-standard.

l_{sn} synchronise les SNODE au niveau des nœuds, et l_{st} synchronise les STREE au niveau des arbres.

La figure suivante montre une S-SSTC entre la phrase anglaise « John picks the heavy box up » et la phrase correspondante en malais « John kutip kota berat itu ».

Les paires de mots se correspondant entre anglais et malais sont : (John, John) , (pick up, kutip), (box, kota), (heavy, berat), (the, itu). Ces correspondance sont enregistrées dans l_{sn} . Puis les correspondances de sous-arbres sont enregistrées dans l_{st}.

[13] « Synchrone » est un terme mal choisi mais historique. En fait, il s'agit plusieurs d'alignement entre deux SSTC.

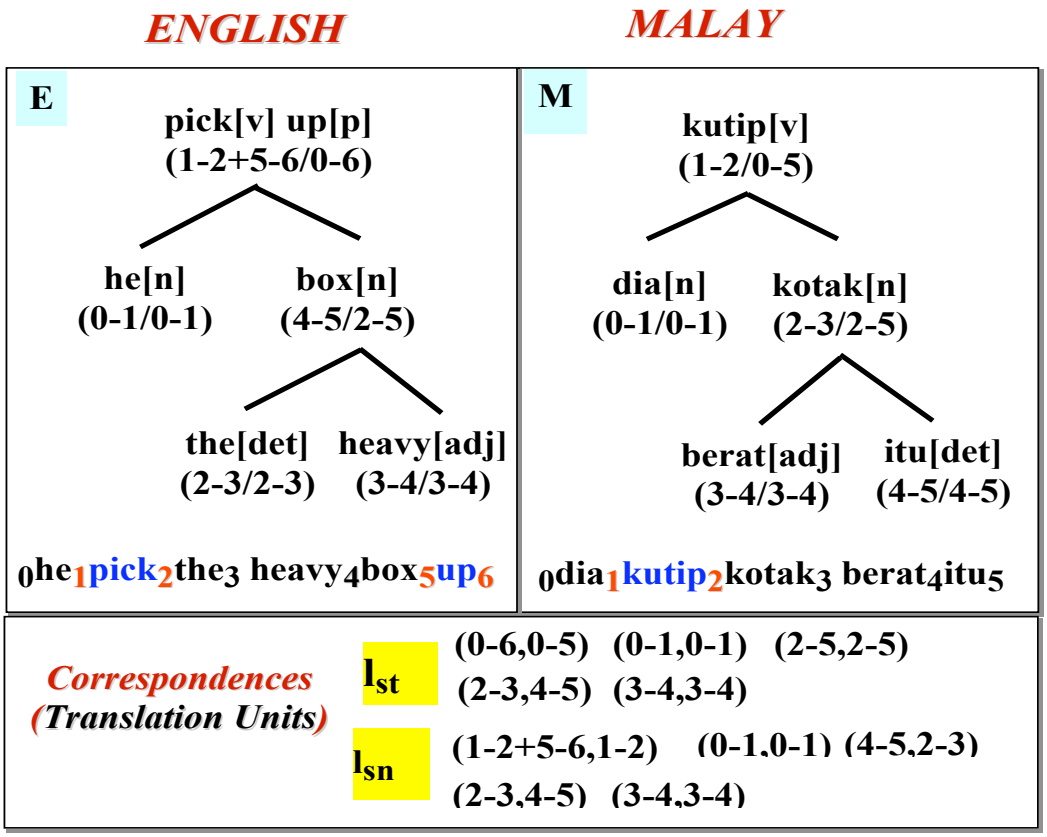


Fig. C-19 Exemple de S-SSTC

Voici maintenant quelques exemples pour montrer la capacité des S-SSTC à représenter des correspondances difficiles entre deux langues.

a) Correspondance « N_1(non-injective) »

```
allemand « Er beschenkte Hans reichlich. »
anglais « He gave Hans an expensive present. »
```

Ici « gave present » correspond à un seul mot « beschenkte » en allemand, et de plus le modifieur « expensive » de « present » devient le modifieur du verbe « beschenkte ».

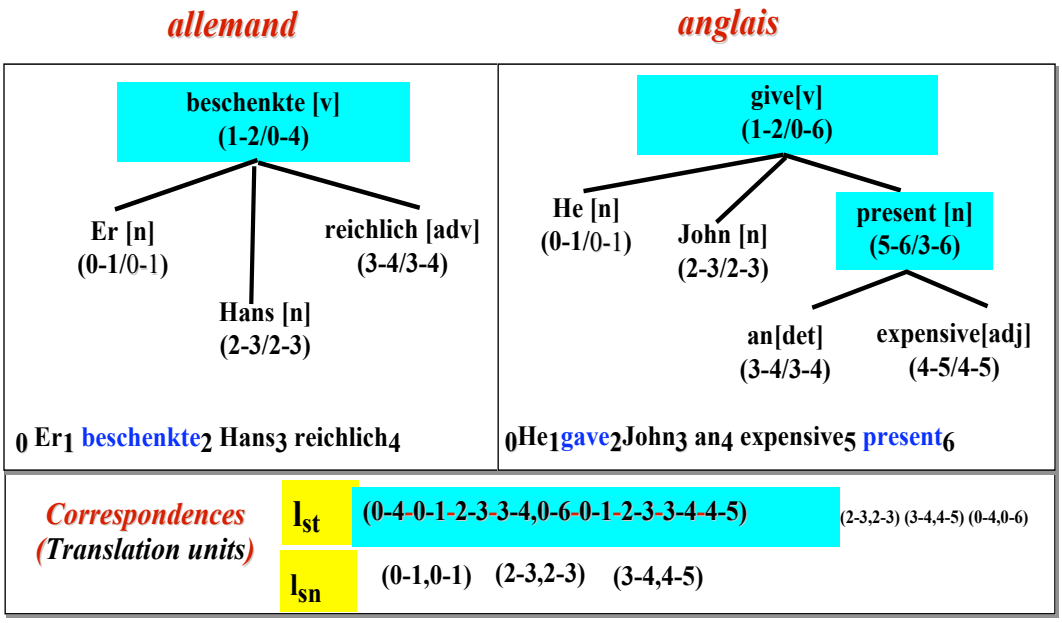


Fig. C-20 S-SSTC pour une correspondance non-injective

b) Inversion de dépendance

```
français « Il monte la rue en courant »
anglais « He runs up the street »
```

Le verbe « monter » en français est exprimé en anglais par une préposition qui devient un dépendant du verbe « run ».

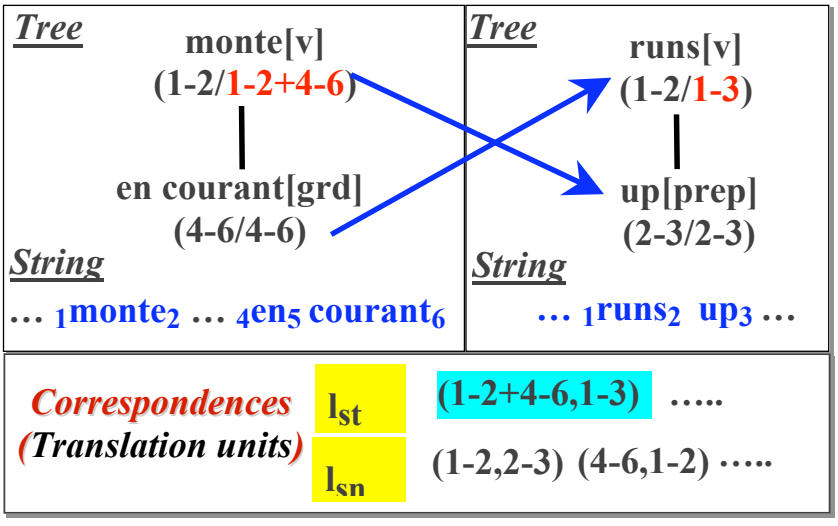


Fig. C-21 S-SSTC pour l'inversion de dépendance

c) Élimination de dépendance

```
français « Le docteur lui soigne les dents »
anglais « The doctor treats his teeth »
```

Ici, en français, le verbe « soigner » a trois compléments (deux arguments et un circonstant, « lui » datif en relation implicite de possesseur avec « dents »). En anglais, le verbe « treat » n'a que deux arguments (comme sujet et objet). Notons que le parallélisme de surface revient dès qu'il n'y a plus de pronom :

```
français « Le docteur soigne les dents de Paul »
anglais « The doctor treats Paul's teeth »
```

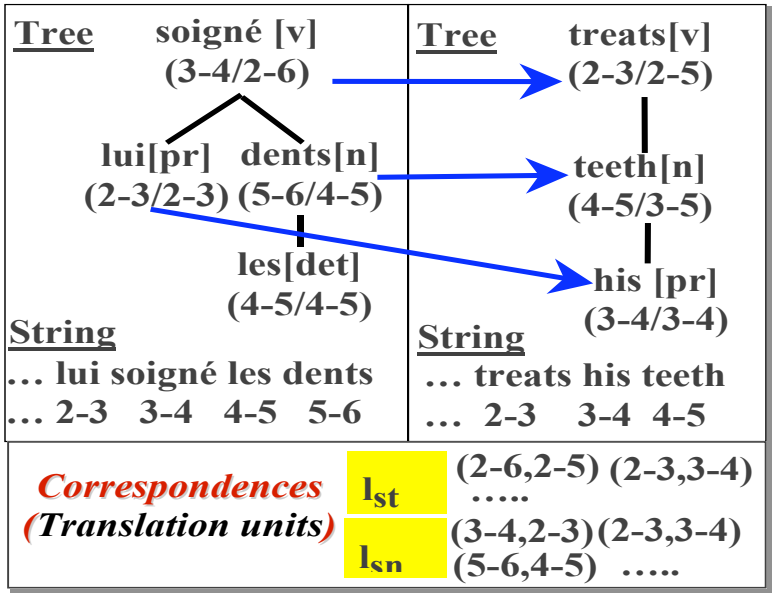


Fig. C-22 S-SSTC pour l'élimination de dépendance

d) Élément discontinu

```
français « Pierre ne l'a pas vu. »
anglais « Pierre has not seen it. »
```

Dans cette correspondance, « not » correspond à « ne.. pas » en français et en plus « ne.. pas » est discontinu.

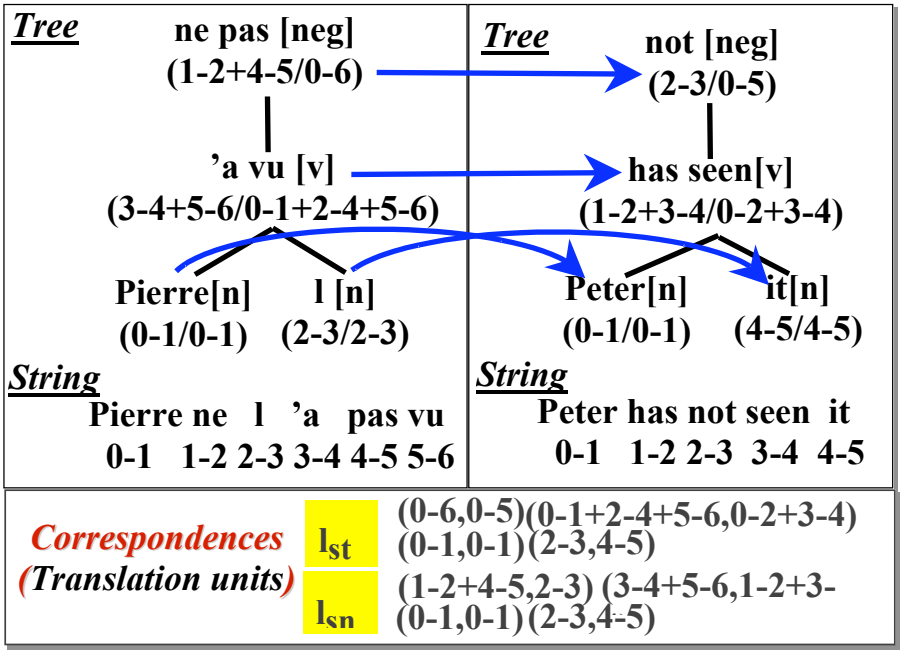


Fig. C-23 S-SSTC pour un élément discontinu

Enfin, nous remarquons qu'une S-SSTC peut spécifier un transfert structural dans les deux sens. Comme elle relie deux SSTC elles-mêmes non orientées, elle permet de spécifier à la fois les 3 étapes d'un système de TA à transfert, et cela dans les deux sens.

En plus, grâce à sa souplesse et à sa finesse, le formalisme des S-SSTC peut être utilisé pour annoter un corpus bilingue. On peut ensuite en extraire des paires de correspondances structurales et lexicales, et c'est cela qui a permis à M. Al-Adhaileh de construire un système de TA réversible à partir d'un corpus anglais-malais annoté par S-SSTC.

Voici comment Al-Adhaileh a réannoté l'exemple fameux suivant, en utilisant une S-SSTC, pour spécifier l'alignement d'un exemple bilingue utilisé par Menezes et Richardson de MSR (Microsoft Research Center) [Menezes 01].

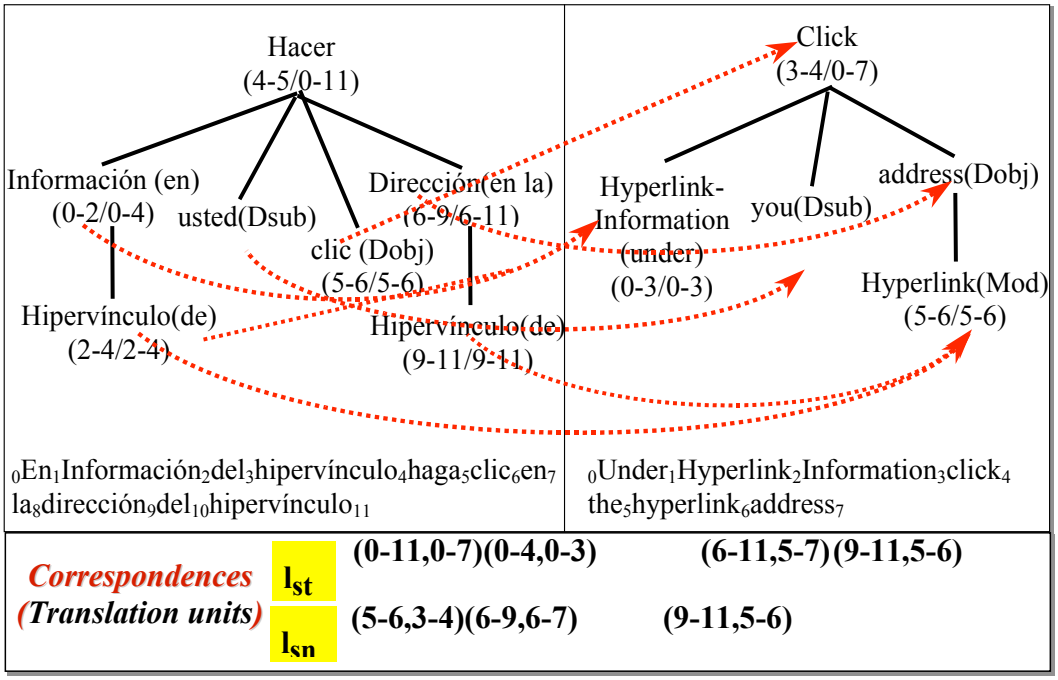


Fig. C-24 S-SSTC d'un exemple de MSR

Donnons quelques détails sur la travail de Tang et Al-Adhaileh (UTMK, USM) sur l'application des S-SSTC à la TA. Dans le cadre de sa thèse, Al-Adhaileh a réalisé un éditeur de S-SSTC [Al-Adhaileh 02], puis il a construit une « base de connaissances bilingue » (BKB) anglais-malais. C'est un corpus bilingue arboré avec les SSTC monolingues et les S-SSTC bilingues. En 2003, il comprenait 40000 exemples, la plupart tirés d'exemples trouvés dans des dictionnaires bilingues.

Ils ont ensuite utilisé ces S-SSTC pour construire un système de TAFE (traduction automatique fondée sur l'exemple).

Voici un écran de l'éditeur de S-SSTC qui montre une synchronisation de SNODE [Chantriaux 03].

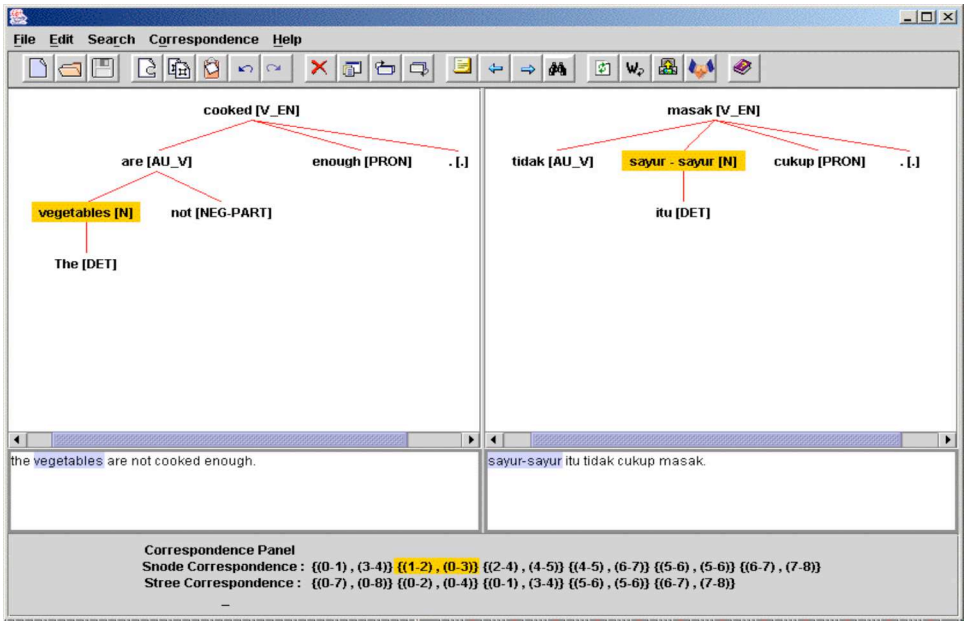


Fig. C-25 Editeur de S-SSTC (I)

Voici un écran montant une synchronisation de STREE :

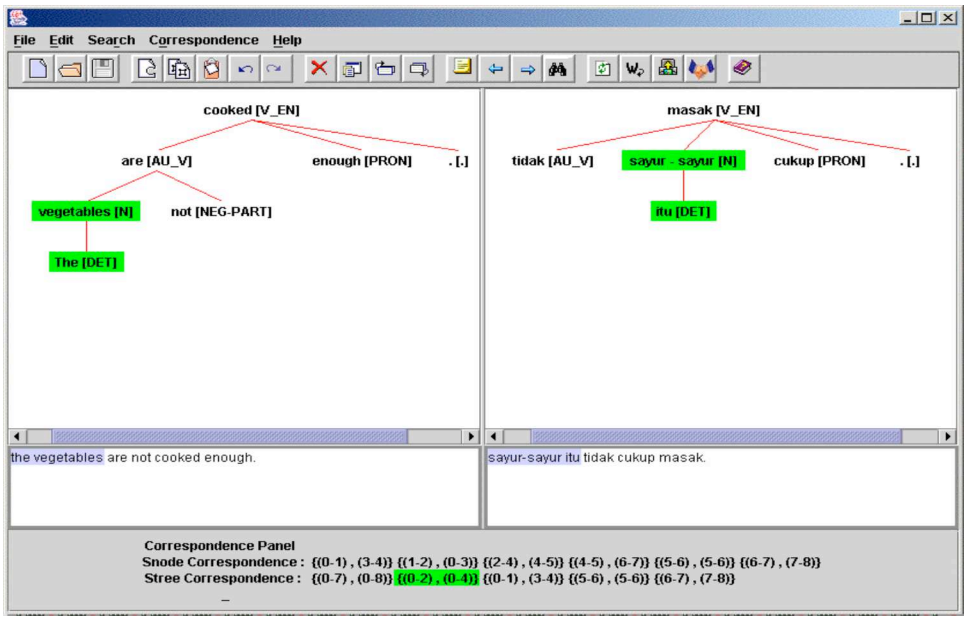


Fig. C-26 Editeur de S-SSTC (II)

1.5 Grammaire Transductive Syntaxique (Sylvain Kahane 2000)

La théorie Sens-Texte (TST) [Mel'_uk 65] considère qu'une langue naturelle est une correspondance entre le sens et le texte. Dans la TST, différents niveaux de représentation ont été définis : morphologique, syntaxique et sémantique (au moins). La figure suivante donne ces 3 représentations pour la phrase anglaise « Peter wants to sell his blue car ». Cependant une langue naturelle n'est pas qu'une correspondance entre deux niveaux de représentation différents, c'est aussi une sous-correspondance (ou « supercorrespondance » selon Kahane dans [Kahane 00]).

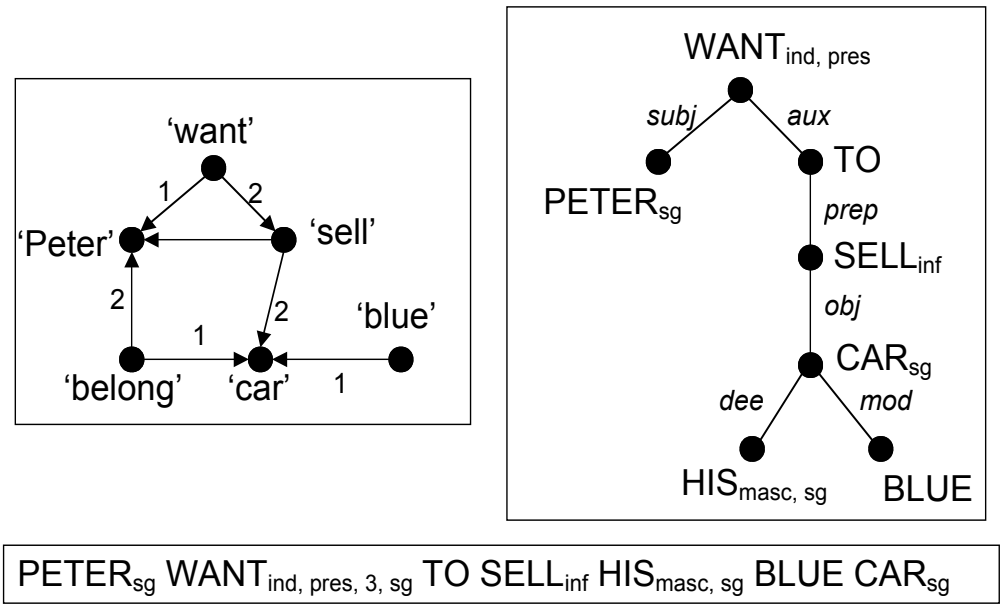


Fig. C-27 Trois niveaux de représentations dans la TST

Dans [Kahane 01], Kahane a proposé un cadre formel pour décrire des modèles Sens-Texte, ou selon le terme de l'auteur, des « grammaires transductives », dont la visée première est de définir une correspondance entre deux ensembles de structures mathématiques, par exemple des suites et des arbres ou des arbres et des graphes.

Une grammaire transductive pour la correspondance entre les niveaux sémantique et syntaxique profond a été proposée par Kahane et Mel'_uk en 1999 [Kahane 99], les structures de représentation étant respectivement des graphes et des arbres de dépendance.

Dans [Kahane 01], Kahane s'intéresse particulièrement à définir une correspondance entre un arbre de dépendance (syntaxique de surface) et une suite (morphologique profonde). Il a donc défini une famille de grammaires transductives *syntaxiques* qu'il a appelée « grammaires de dépendance atomiques ». Ces grammaires sont atomiques dans le sens où elles mettent en relation uniquement des atomes de structure, c'est-à-dire des nœuds ou des arcs.

Les règles sont donc de deux types : les règles sagittales (en latin sagitta = flèche), qui mettent en relation une dépendance entre deux nœuds avec une relation d'ordre entre deux nœuds, et les règles nodales, qui mettent en relation un nœud avec un nœud.

La raison pour laquelle le modèle syntaxique est choisi est que c'est le modèle qui a le plus d'intérêt parmi tous les modèles de la TST.

Voici la définition formelle de la grammaire de dépendance atomique qu'on peut trouver dans [Kahane 00] :

> Une grammaire de dépendance atomique est un quintuplet G=(_,C,R,O,_) où _ est l'ensemble des lexies, C est l'ensemble des catégories grammaticales, R est l'ensemble des relations syntaxiques, O est l'ensemble des positions linéaires et _ est l'ensemble des règles sagittales, à savoir un sous-ensemble de RxOxCxC.

L'exemple suivant nous donne une idée plus claire de la définition de cette grammaire. Nous y voyons deux structures de la phrase anglaise « Peter eats red beans » : syntaxique et morphologique profonde.

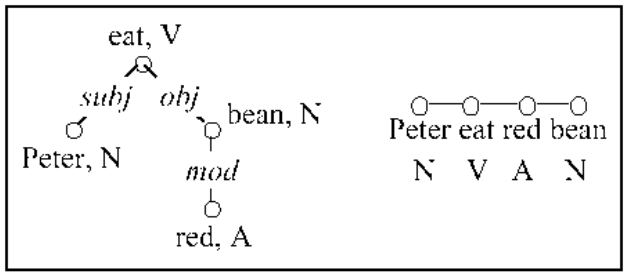


Fig. C-28 Deux structures de « Peter eats red beans »

Une grammaire de dépendance atomique mettant en correspondance ces deux structures est la suivante :

Soit $G_0 = (\__0 , C_0 , R_0 , O_0 , \__0)$ avec :
- $\__0$ = {Peter, bean, eat, red}
- C_0 = {V, N, A}
- R_0 = {subj, obj, mod}
- O_0 = {<, >}
- $\__0$ = {(subj, <, V, N), (obj, >, V, N), (mod, <, N, A)}

Il faut d'abord noter que l'ensemble de catégories et de relations (C_0 et R_0) est ici limité et est seulement suffisant pour décrire cet exemple. Une grammaire de dépendance atomique avec O_0 = {<, >} permet seulement de spécifier la position d'un nœud par rapport à son gouverneur, à savoir avant (<) ou après (>).

La règle sagittale (subj, <, V, N) signifie que, pour chaque couple de lexies X et Y tel que X est un V(erbe) et Y est un N(om), la dépendance syntaxique X-subj_Y respectera l'ordre linéaire Y<X (Y est avant X).

Dans les graphes suivants, les flèches notent la correspondance entre deux morceaux de structure. La flèche _ veut dire qu'il est possible d'appliquer cette règle en synthèse et la flèche _ veut dire qu'il est possible d'appliquer cette règle en analyse. La flèche _, comme dans la figure suivante, indique que la règle marche dans les deux sens.

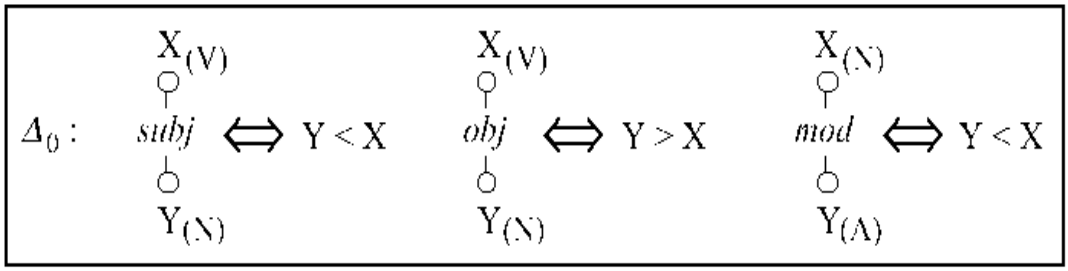


Fig. C-29 Règles de _0 dans le style de la TST

La figure suivante montre comment G_0 peut être utilisée comme grammaire transductive en synthèse. Les règles de $\__0$ s'appliquent sur toutes les branches de l'arbre T et ainsi on obtient une séquence. Pour obtenir toutes les séquences correspondant à un arbre T, toutes les combinaisons de règles doivent être essayées.

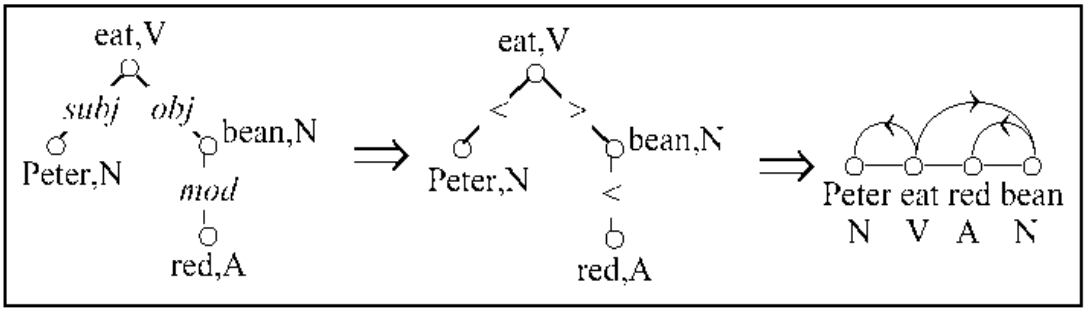


Fig. C-30 G_0 utilisée comme grammaire transductive en synthèse

La figure suivante montre la procédure d'analyse utilisant G_0. C'est la procédure inverse de la synthèse.

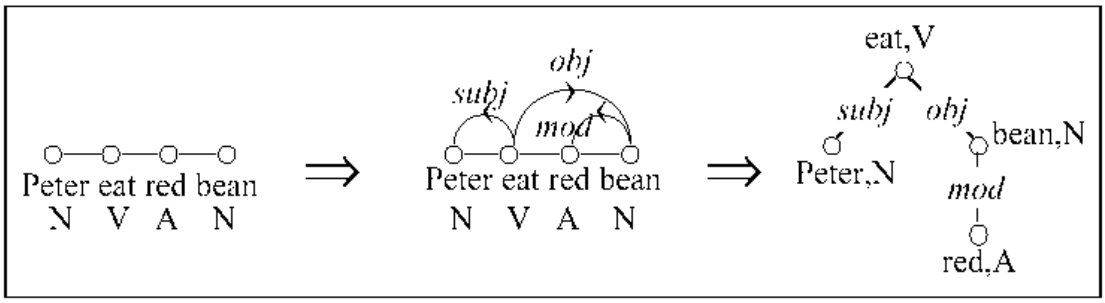


Fig. C-31 G_0 utilisée comme grammaire transductive en analyse

Les procédures d'analyse et de synthèse peuvent paraître très simples, voire trop simples. Mais il y a d'autres règles pour affiner et obtenir un meilleur résultat.

A part les études sur les correspondances chaîne-arbre, il y a aussi des études sur les correspondances entre deux structures de même type, par exemple, entre arbre et arbre.

Ainsi, la « Pattern-Based Translation » de Takeda [Takeda 90] repose sur des patrons de structures CFG (Context Free Grammar) servant de pont entre deux langues.

Voici trois patrons pour définir la correspondance entre la phrase anglaise « *Pronom* take a look at *Nom* » et la japonaise « *Pronom* wa *Nom* wo miru », et l'application de ces patrons aux deux arbres.

(p1) take:VERB:1 a look at NP2⇒VP:1
VP:1⇐NP:2 wo(dobj) miru(see):VERB:1
(p2) NP:1 VP:2 ⇒ S:2 S:2 ⇐ NP:1 ha VP:2
(p3) PRON:1 ⇒ NP:1 NP:1 ⇐ PRON:1

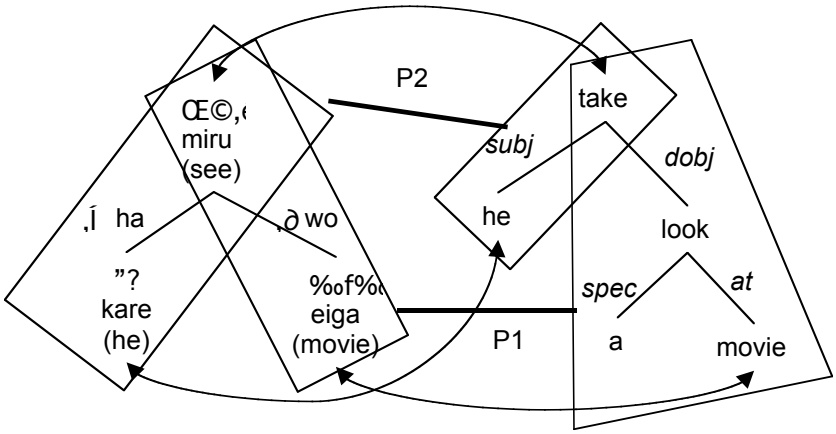
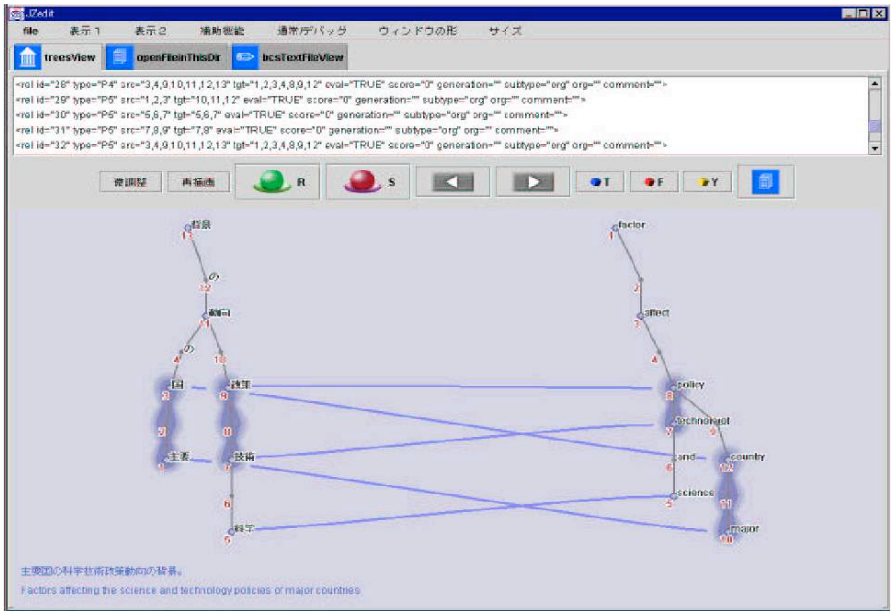
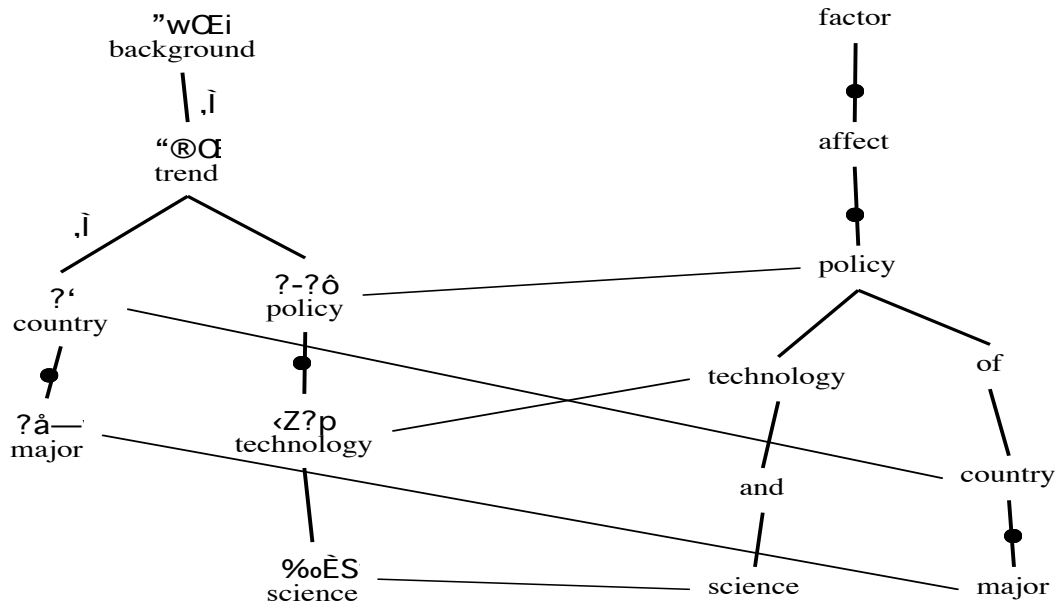


Fig. C-32 Trois patrons dans la « Pattern-Based Translation » de Takeda

Watanabe [Watanabe 00] chez IBM Japan, Menezes et Richardson chez Microsoft [Menezes 01] ont aussi beaucoup étudié l'extraction de correspondances de sous-arbres à partir de corpus bilingues et parsés. Voici une interface de l'outil de Watanabe pour manipuler et présenter la correspondance entre deux arbres. Nous redéssinons en plus bas ces deux arbres correspondantes.





?å—v?‘,Ì%‰ÈSw‹Z?p?-?ô“®Œü,Ì
Factors affecting the science and technologi policies of major countries

Fig. C-33 Interface de Watanabe pour présenter la correspondance entre deux arbres

2. Étude des correspondances UNL-énoncé dans les corpus disponibles

Introduction

Nous n'avons trouvé aucune étude formelle sur les correspondances entre graphes et chaînes nous permettant de modéliser directement la correspondance entre un hypergraphe UNL et un énoncé correspondant. D'autre part, nous avons vu comment obtenir une telle description formelle en considérant qu'une correspondance graphe-texte pourrait être décomposée en un produit de trois correspondances, à l'aide de deux structures intermédiaires, un arbre et un treillis.

Il est toutefois intéressant de mener une étude expérimentale pour chercher à déterminer quels types de correspondances on trouve en pratique entre les graphes UNL et les énoncés correspondants dans les différentes langues.

Nous présentons d'abord les corpus UNL dont nous disposons, leurs statuts et leurs contenus. Puis nous définissons les hiérarchies du graphe UNL et du treillis de texte, et montrons ensuite les correspondances graphe-texte que nous avons identifiées.

2.1 Présentation des corpus

Voici d'abord la liste des balises et abréviations utilisées.

s :	nombre de phrases	org :	langue source	unl :	graphe UNL
xml :	xml-isé	date :	date du corpus		
ab :	arabe	cn :	chinois	de :	allemand
el :	anglais	es :	espagnol	fr :	français
id :	indonésien	hd :	hindi	it :	italien
jo :	arabe jordanien	jp :	japonais	lv :	letton
mg :	mongol	pg :	portugais	ru :	russe
sh :	swahili	th :	thaï		

Voici ensuite la liste des corpus que nous avons collectés. Dans la colonne la plus à gauche, ce sont les noms de corpus. Si la version de la langue existe dans ce corpus, la cellule correspondante contient une étoile. Il arrive parfois qu'il manque quelques phrases pour certaines langues. Dans ce cas, nous utilisons un chiffre pour représenter le nombre de phrases qui ont une version dans la langue de la colonne.

Tous les corpus nous sont parvenus sous forme de documents UNL-html, et nous avons transformé en forme UNL-xml plusieurs corpus qui sont fortement multilingues.

	s	**org**	**unl**	**ab**	**cn**	**de**	**el**	**es**	**fr**	**id**	**hd**	**it**	**jo**	**jp**	**lv**	**mg**	**pg**	**ru**	**sh**	**th**	**xml**	**date**
UNESCO	50	el	*				*		*													11/03
UNL-HEREIN	23	es	*					*														06/03
La main a la pâte	10	fr	*				*	*	*			*						*			*	04/03
Sevres	10	fr	2		*	*	*	*	*								*				*	01/03
UNL news																						
UNL news 1	7	el	*	*	*		*	*	*	*	*	*		*				*			*	02/02
UNL news2	10	el	*				*															05/02
UNL news3	21	el	*				*															07/02
FB2004																						
FB2004-I	30	el	*				*	*	*		*	*						*			*	04/01
FB2004-II	92	el	*				*	*										*			*	04/01
geneve 2001																						
aral.xml	16	es	*					*				*									*	01/01
oper.xml	21	ru	*				*	*				*						*			*	01/01
ultra5.xml	22	it	*					*				*									*	01/01
org-explorer	322	el	*				*							*								02/00
org-information	19	el	*	*	*	*	*	*		*	*	*	*	*	*		*	*		*	*	02/00
love	14	el	*				*		*													10/99
babel tower	30	el	*				*															12/96

Tableau C-1 Corpus UNL traités

Ces corpus ne sont pas homogènes en qualité, codage et versions linguistiques. Il existe des versions linguistiques qui ont été produites par des humains et pas par

déconversion, ou qui ont été traduites par machine depuis l'anglais au lieu d'UNL. Il existe aussi des corpus où manquent certaines versions linguistiques de certaines phrases.

D'autre part, les fichiers de ces corpus sont des fichiers texte, où les codages sont différents d'une langue à l'autre (et parfois pour une même langue). Il n'est pas toujours facile de les transformer en Unicode et en XML.

Voici quelques indications sur chaque corpus et des exemples de textes.

2.1.1 Babel Tower

Babel Tower est l'un des plus anciens corpus (décembre 1996). Il a été repris plus tard dans un séminaire UNL en juin 1999 à Pérouge. Il comprend 30 phrases et son contenu est une introduction aux langues du monde, qui parle de la nécessité d'un langage pour traverser la barrière des langues, en citant l'histoire biblique de la tour de Babel. Nous n'avons pas trouvé d'autres versions de langues que l'anglais, le français et le graphe UNL. Ce corpus n'a pas beaucoup d'intérêt parce que la Bible est déjà le document le plus traduit dans le monde.

Voici un texte de ce corpus :
[S]
;<BAB1>
;The Tower of Babel
mod(tower(icl>building).@entry.@title.@def,babel(icl>place name))
[/S]
Serveur de développement :
La tour d'un? <babel>.
[S]
;<BAB2>
;Long ago, in the city of Babylon, the people began to build a huge tower, which seemed about to reach the heavens.
tim(begin(icl>do(obj>thing)).@entry.@pred.@past,long ago)
mod(city(icl>region).@def,babylon(icl>country))
plc(begin(icl>do(obj>thing)).@entry.@pred.@past,city(icl>region).@def)
agt(begin(icl>do(obj>thing)).@entry.@pred.@past,people(icl>person).@def)
obj(begin(icl>do(obj>thing)).@entry.@pred.@past,build(icl>construct).@pred)
agt(build(icl>construct).@pred,people(icl>person).@def)
obj(build(icl>construct).@pred,tower(icl>building).@indef)
aoj(huge(aoj>thing),tower(icl>building).@indef)
aoj(seem(aoj>person,obj>thing).@pred.@past,tower(icl>building).@indef)
obj(seem(aoj>person,obj>thing).@pred.@past,reach(icl>do(gol>thing)).@pred.@begin-soon)
obj(reach(icl>do(gol>thing)).@pred.@begin-soon,tower(icl>building).@indef)
gol(reach(icl>do(gol>thing)).@pred.@begin-soon,heaven(icl>region).@def.@pl)
[/S]
Serveur :
Le peuple a commencé à construire une énorme tour a semblé qu'elle est atteinte les cieux dans la cité d'une Babylone jadis

Nous pouvons constater que, le projet étant à peine commencé, on n'avait pas encore bien maîtrisé l'usage des balises UNL. On ne faisait pas attention, et il manque des balises pour marquer la version de langue. Aussi, les spécifications n'étaient pas tout à fait les mêmes qu'aujourd'hui. On utilisait la restriction « .@pred » qu'on n'utilise plus.

Notons que la traduction est fausse parce que le graphe est douteux : « tower(icl>building) » est analysé comme objet (obj) de

« reach(icl>do(gol>thing)) », alors que ce devrait être l'agent (agt), puisque la restriction « icl>do » sur reach indique un verbe transitif anglais. Il est vrai que les spécifications réservent agt pour « agent volitif ». On devrait alors avoir « icl>occur ». Il y a une autre erreur (« a semblé qu'elle » au lieu de « qui semblait »), sans doute imputable à la déconversion et corrigeable.

Cet exemple montre bien la nécessité de la normalisation de l'usage d'UNL, telle qu'elle est faite (et progresse) depuis le projet FB2004 (voir plus bas).

2.1.2 Love

LOVE est un corpus de 14 phrases courtes parlant d'amour. Le français a été produit par le déconvertisseur du GETA, à l'automne 1999. A la même époque, il y avait aussi les corpus great.unl, plan.unl, etc. Ces corpus ont été produits par le centre UNL comme un exercice de déconversion.

Voici un extrait de ce corpus :

```
[S]
;<LOVE_01>
agt(adore(icl>love).@entry.@present,he)
obj(adore(icl>love).@entry.@present,brother(icl>kinfolk))
pos(brother(icl>kinfolk),he)
mod(brother(icl>kinfolk),elder(mod>person))
[/S]
{el} he adores his elder brother {/el}
{fr} il adore son frère ainé {/fr}

[S]
;<LOVE_02>
obj(be beloved(icl>love).@entry.@present,she.@topic)
agt(be
beloved(icl>love).@entry.@present,friend(icl>comrade).@pl)
pos(friend(icl>comrade).@pl,she)
mod(friend(icl>comrade).@pl,all)
qua(friend(icl>comrade).@pl,many)
mod(many,many)
[/S]
[el] She is beloved by all her many, many friends.{/el}
{fr} tous ses nombreux amis l'aiment {/fr}
```

Nous pouvons constater que l'encodage du graphe UNL n'était pas assez sophistiqué. Ainsi, la deuxième phrase a été codée selon la syntaxe de surface de l'anglais.

2.1.3 Sport

Sport est un corpus préparé pour une démonstration au Symposium UNL qui a eu lieu aux Nations Unies, à New York, en novembre 1998. Comme c'était juste après la coup du monde de football (gagnée par la France), on avait choisi des phrases dans des articles de presse sur le sujet, dans différentes langues.

Voici un extrait de ce corpus :

```
[S]
;Play restarts when the ball touches the ground.
```

```
obj(restart(icl>begin,man>anew,obj>process).@pred.@entry,play(
fld>sport,icl>period))
tim(restart(icl>begin,man>anew,obj>process).@pred.@entry,touch
(cob>thing,icl>contact,man>physically,obj>thing).@pred)
agt(touch(cob>thing,icl>contact,man>physically,obj>thing).@pre
d,ball(fld>soccer,icl>tool).@def)
obj(touch(cob>thing,icl>contact,man>physically,obj>thing).@pre
d,ground(icl>place).@def)
[/S]
{fr} Redemarre un jeu[14] quand le ballon touche le sol.{/fr}

[S]
;The ball is dropped again: if it is touched by a player
before it makes contact with the ground.
obj(drop(agt>person,icl>#event,obj>thing).@pred.@entry,ball(fl
d>soccer,icl>tool):01.@theme.@def)
man(drop(agt>person,icl>#event,obj>thing).@pred.@entry,again(i
cl>once more))
con(drop(agt>person,icl>#event,obj>thing).@pred.@entry,touch(a
gt>part of
body,icl>contact,man>physically,obj>thing):01.@pred.@past.@com
plete)
agt(touch(agt>part of body, icl>contact, man>physically,
obj>thing):01.@pred.@past.@complete, player(fld>soccer,
icl>sportsman))
obj(touch(agt>part of
body,icl>contact,man>physically,obj>thing):01.@pred.@past.@com
plete,ball(fld>soccer,icl>tool):03.@def)
tim(touch(agt>part of
body,icl>contact,man>physically,obj>thing):01.@pred.@past.@com
plete,before(obj>time))
obj(before(obj>time),touch(cob>thing,icl>contact,man>physicall
y,obj>thing):02.@pred.@past.@complete)
agt(touch(cob>thing,icl>contact,man>physically,obj>thing):02.@
pred.@past.@complete,ball(fld>soccer,icl>tool):02.@def)
obj(touch(cob>thing,icl>contact,man>physically,obj>thing):02.@
pred.@past.@complete,ground(icl>place).@def)
[/S]
{fr} Le ballon est encore lance si un joueur a avant que le
ballon ait touche le sol touche le ballon. {/fr}
```

Nous remarquons que les restrictions dans ce corpus sont plutôt précises et compliquées. Dans les spécifications ultérieures, les types de relation décrivant la hiérarchie ont été réduits à trois : icl, pof, mod. En plus, la hiérarchie de la KB a été changée plusieurs fois au cours du développement du projet. Donc, nous ne trouvons plus les UW ci-dessus dans la KB d'aujourd'hui. Il est probable que cette partie de KB a été produite exprès pour cette démonstration.

C'est aussi la première fois que le centre UNL a commencé à organiser des démonstrations multilingues pour promouvoir UNL.

[14] On a « un jeu » car « play(fld>sport, icl>person) » n'a pas l'attribut .@def, au contraire de « ground(icl>place).@def ».

2.1.4 Org-Explorer

Org-Explorer est un corpus qui comprend 322 phrases. Il a été créé pour montrer les fonctionnalités d'Org-explorer, qui est une application proposée par le centre UNL pour montrer le multilinguisme d'UNL. Le contenu est l'introduction à l'ONU, et présente sa hiérarchie, les responsables et les fonctionnalités de tous les départements en plusieurs langues.

19 phrases sont vraiment multilingues, avec des versions en 14 langues. Les autres phrases sont au moins en japonais et en anglais.

Nous avons réuni ces 19 phrases dans un autre corpus qui s'appelle **Org-Information**. Les phrases n'y sont donc pas reliées l'une à l'autre. Selon l'auteur, il y a des versions linguistiques qui ont vraiment été produites par déconversion, et d'autres non.

Voici une image de la conception de cet Org-Explorer :

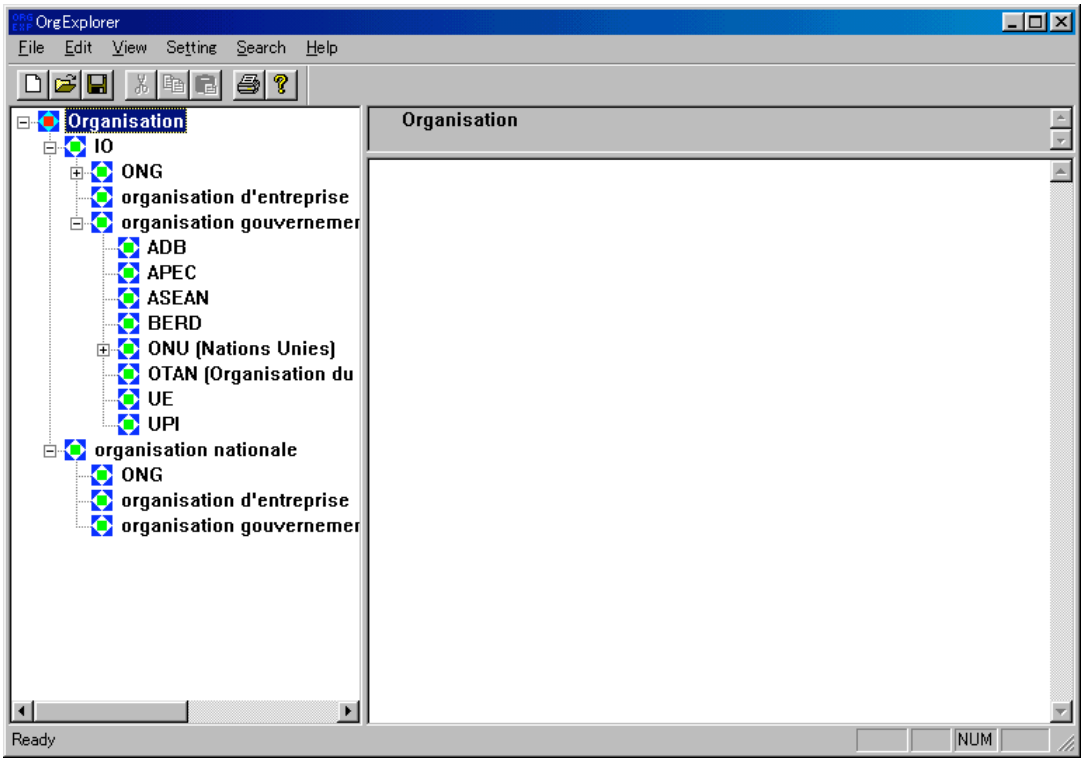


Fig. C-34 Structure d'Org-Explorer

Ce corpus comprend plusieurs codages. Voici une partie de ce corpus sous Notepad. Notons qu'il n'existe pas d'outil qui permette de visualiser tous les caractères de ces codages. C'est une des raisons qui nous ont poussé à définir le format UNL-xml et à imposer la normalisation du codage dans ce format (Unicode/UTF-8).

Fig. C-35 Org-Information sous Notepad

Ce corpus a donc été transformé par nos soins en UNL-xml. Voici la même partie de corpus. Avec Unicode, nous pouvons voir toutes les versions.

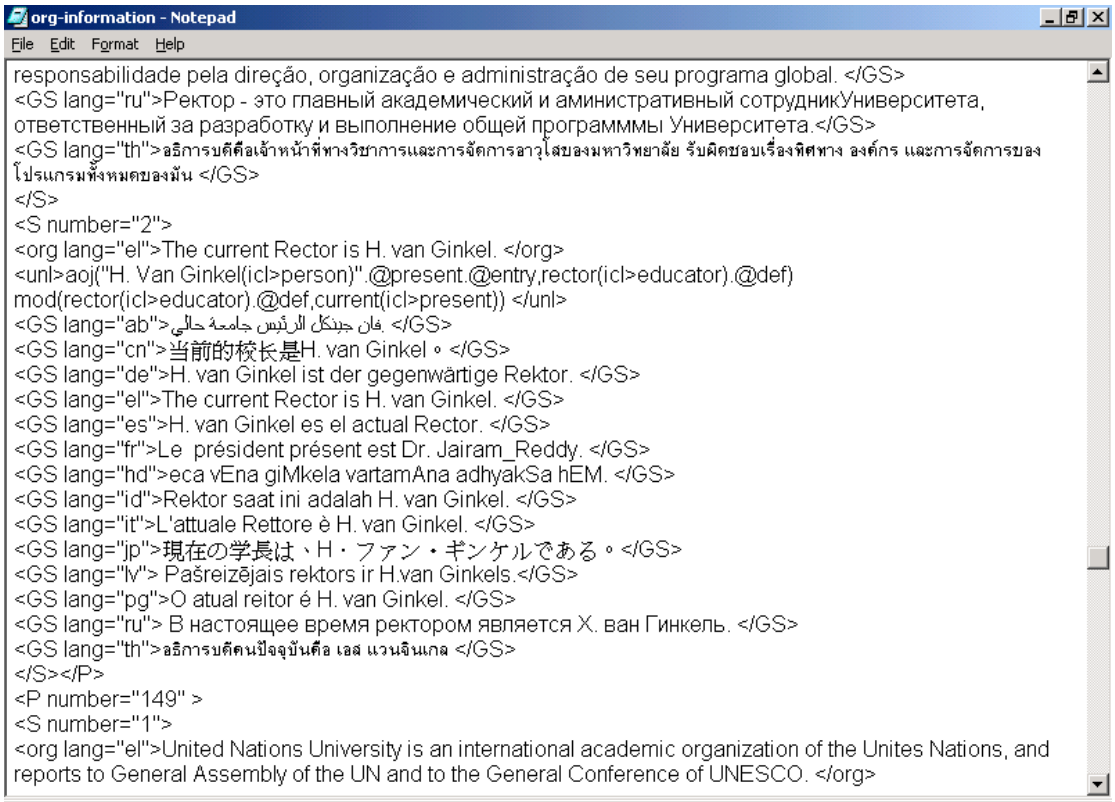


Fig. C-36 Corpus Org-Information en format UNL-xml sous Notepad

Plusieurs équipes ont tenté de montrer correctement le texte en évitant les caractères spéciaux ou les accents. Par exemple, les équipe allemande et italienne n'utilisent pas du tout les lettres accentuées. Les Umlaut en allemand ont été remplacés par le tréma, par exemple, ü _u". Les accents graves ont été remplacés par une apostrophe, è _e'. Pour tous les corpus, l'équipe indienne abandonne l'alphabet dévanagari et utilise l'alphabet romain.

Avec ce corpus, plein de noms propres et de noms d'organisations (par exemple, United Nations University, UNU Programme for Biotechnology in Latin America and the Caribbean, Conference of Directors, UNU Institute for Software Technology, etc.), on a commencé à identifier le problème du nom propre (ou nom composé). Le nombre de noms propres de ce genre est illimité et donc on ne peut pas créer pour chacun une UW. Il faut avoir un algorithme pour calculer ces noms. Plusieurs solutions possibles ont été proposées dans [Boitet 02d] et [Boguslavsky 02a, 02b].

2.1.5 Genève 2001

Genève 2001 comprend trois articles choisis séparément par les équipes russe, italienne et espagnole, enconvertis manuellement, puis déconvertis dans ces trois langues. Il s'agissait de montrer la qualité de la déconversion. Cet effort a été organisé par le centre espagnol pendant le symposium UNL 2001 à Genève. Toutes les phrases ont été produites par les déconvertisseurs. Les trois articles comprennent au total 59 phrases.

Voici un extrait de ce corpus.

```
[D:dn=Mar Aral version final,on=UNL Spain,
mid=carde@opera.dia.fi.upm.es]
[P:1]
[S:1]
{org:es}
El mar Aral, situado entre las repúblicas de Uzbekistán y
Kazajstán, era el cuarto mar interior más grande del mundo.
{/org}
{unl}
nam(sea:01.@def, Aral)
obj(locate(icl>do).@present, sea:01.@def)
man(locate(icl>do).@present, between(icl>manner))
obj(between(icl>manner), republic:01.@def)
and(republic:01.@def, republic:02.@def)
nam(republic:01.@def, Uzbekistan)
nam(republic:02.@def, Kazakhstan)
aoj(sea:02.@def.@entry.@past, sea:01.@def)
mod(sea:02.@def.@entry.@past, inland(mod<thing))
mod(sea:02.@def.@entry.@past, fourth(mod<thing))
mod(sea:02.@def.@entry.@past, large)
man(large, most)
scn(large, world.@def)
{/unl}
{es}
El mar Aral que es ubicado la república de Uzbekistán y la
república Kazajstán era el cuarto mar más grande en el mundo
interior.
{/es}
```

```
{it}
Il mare Aral che e' localizzato tra la repubblica Uzbekistan e
la repubblica Kazakhstan e' stato il quarto mare piu' vasto
nel mondo interno. {/it}
{ru}
_________ ____, _______ ____________ _____ ___________
_________ _ ___________ __________, ____ _________ ________
_______ _ ____ __________ ____.
{/ru}
[/S]
```

Le corpus lui-même respecte les spécifications d'un document UNL-html.

C'est la première fois que les participants du projet UNL organisaient spontanément une démonstration d'UNL sans la direction du centre UNL. On constate que les restrictions des UW sont plus simples, car elles ne réfèrent pas à la KB.

2.1.6 UNL News

UNL News contient des nouvelles publiées par le centre UNL dans une sorte de journal électronique, pour communiquer et faire de la publicité sur Internet. Chaque numéro de ce journal est stocké sur le site du centre UNL. Avec UNL-viewer, l'utilisateur peut choisir la version de langue de lecture. Le centre UNL voulait le publier régulièrement et en autant de langues que possible, mais n'a publié que trois numéros jusqu'à présent. Le premier numéro est le plus complet, et nous nous en servons beaucoup comme exemple d'un vrai document multilingue UNL-xml.

On trouve le fichier complet de ce corpus dans l'Annexe C.

Voici la page web d'UNL news :

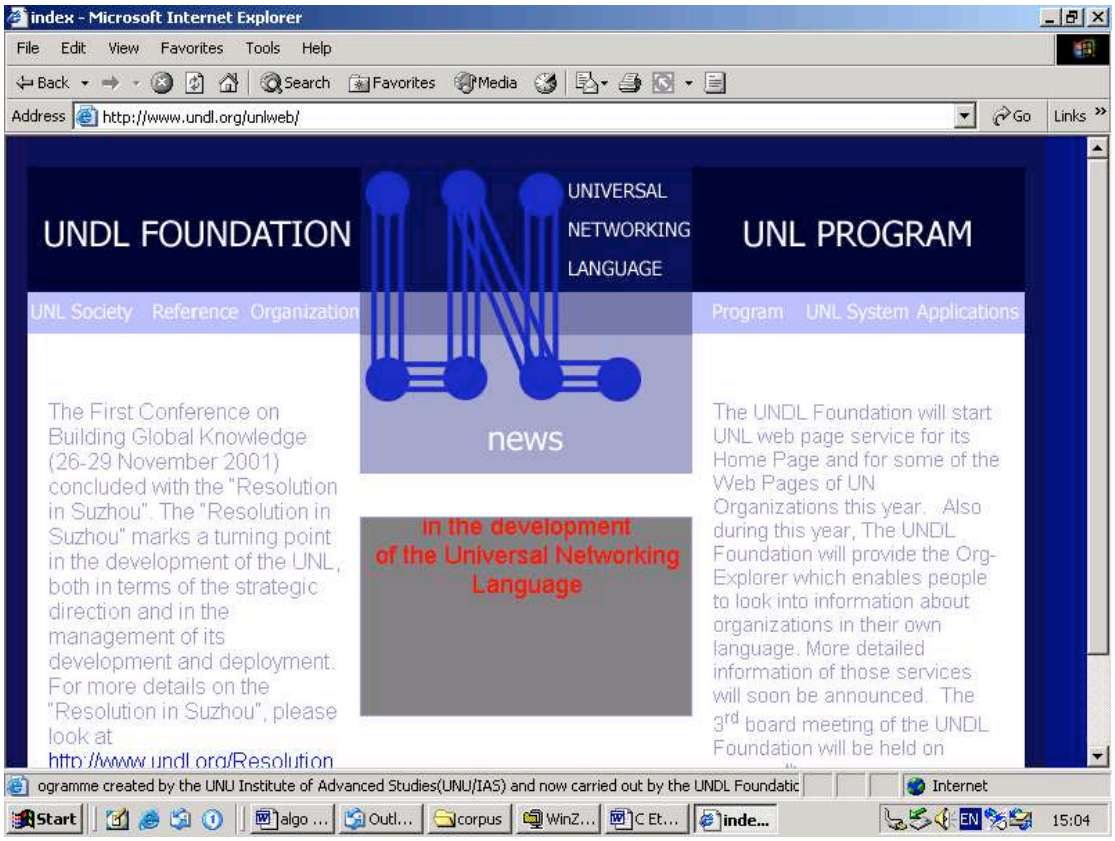


Fig. C-37 Page d'accueil de UNL News

Voici un extrait de ce corpus.

```
[S:3]
{org:el}
The First Conference on Building Global Knowledge (26-29
November 2001) concluded with the "Resolution in Suzhou".
{/org}
{unl}
obj(conclude(icl>end(obj>thing)).@entry.@past, :01)
mod:01(conference(icl>meeting).@entry.@def, 1.@ordinal)
aoj:01(on(icl>about), conference(icl>meeting).@entry.@def)
obj:01(on(icl>about), build(agt>thing,obj>thing))
obj:01(build(agt>thing,obj>thing), knowledge(icl>information))
mod:01(knowledge(icl>information), global(icl>worldwide))
tim(:01, day(icl>date).@pl)
tim(November, year(icl>date))
tim(day(icl>date).@pl, November)
mod(year(icl>date), 2001)
mod(day(icl>date).@pl, :02)
fmt:02(26.@entry,29)
man(conclude(icl>end(obj>thing)).@entry.@past, with(icl>how
      (obj>thing)))
obj(with(icl>how(obj>thing)), :03.@double_quote)
plc:03(resolution(icl>decision).@entry.@def, Suzhou(icl>city))
{/unl}
{ab}
_______ _____ __ ____ _____ ______ (26-29 ______ 2001) _____
_"_______ __ ____
```

```
{/ab}
{cn}
_____________(26-29 2001  11_) _" ___Suzhou " ___
{/cn}
{el}
The First Conference on Building Global Knowledge (26-29
November 2001) concluded with the "Resolution in Suzhou".
{/el}
{es}
la conferencia en saber global se construye de 1 en días en
noviembre en año de 2001 de 26 a 29 se concluyó con la
resolución en Suzhou.
{/es}
{fr}
La première conférence sur la construction de connaissance
globale les jours 26-29 novembre 2001 s'est conclue avec la
"résolution à Suzhou".
{/fr}
{hd}
sArvaBOmika jFAna nirmANa para pahalA sammelana (26-29
navaMbara 2001) "sujZU prastAva" ke sAtha samApta huA.
{/hd}
{id}
Konferensi pertama  mengenai membangun pengetahuan global
pada hari-hari  dari 26 sampai 29 Nopember tahun 2001 telah
menyimpulkan resolusi di Suzhou.
{/id}
{it}
La Prima Conferenza sulla Creazione della Conoscenza Globale
nei giorni 26-29 Novembre 2001 si e' conclusa con la
"Risoluzione di Suzhou".
{/it}
{jp}
_____________________________“___
___”_______{/jp}
{ru}
______ ___________ «__________ __________ ______» (26 - 29
______ 2001) ___________ _________ «_________ _ ______».
{/ru}
[/S]
```

Ce corpus comprend 10 langues différentes. Certaines versions ont été produites par déconversion et les autres manuellement.

Seul UNL News1 a été XML-isé, parce que les deux autres n'ont pas encore été déconvertis vers d'autres langues. En effet, les développeurs se sont mis à travailler sur des corpus UNL provenant de textes non produits par le projet.

2.1.7 FB2004

FB2004 est un corpus produit par le projet FB2004. Il y a environ 2800 mots et 122 phrases dans ce corpus. Comme nous l'avons dit plus haut (section B.2.3.4.2), il y a eu deux phases dans ce projet. Dans la première phase, 30 phrases ont été encodées en six langues (espagnol, anglais, français, italien, russe et hindi). Dans la deuxième phase, 92 phases ont été encodées en trois langues (espagnol, russe et anglais). Toutes

les phrases de ce corpus, sauf la version anglaise, proviennent des déconvertisseurs. C'est un corpus très précieux, parce qu'il nous montre la qualité que les déconvertisseurs peuvent atteindre, après quelques réglages des graphes et des déconvertisseurs.

Tous les résultats et les procédures sont stockés sur le site FB2004. Voici sa page d'accueil :

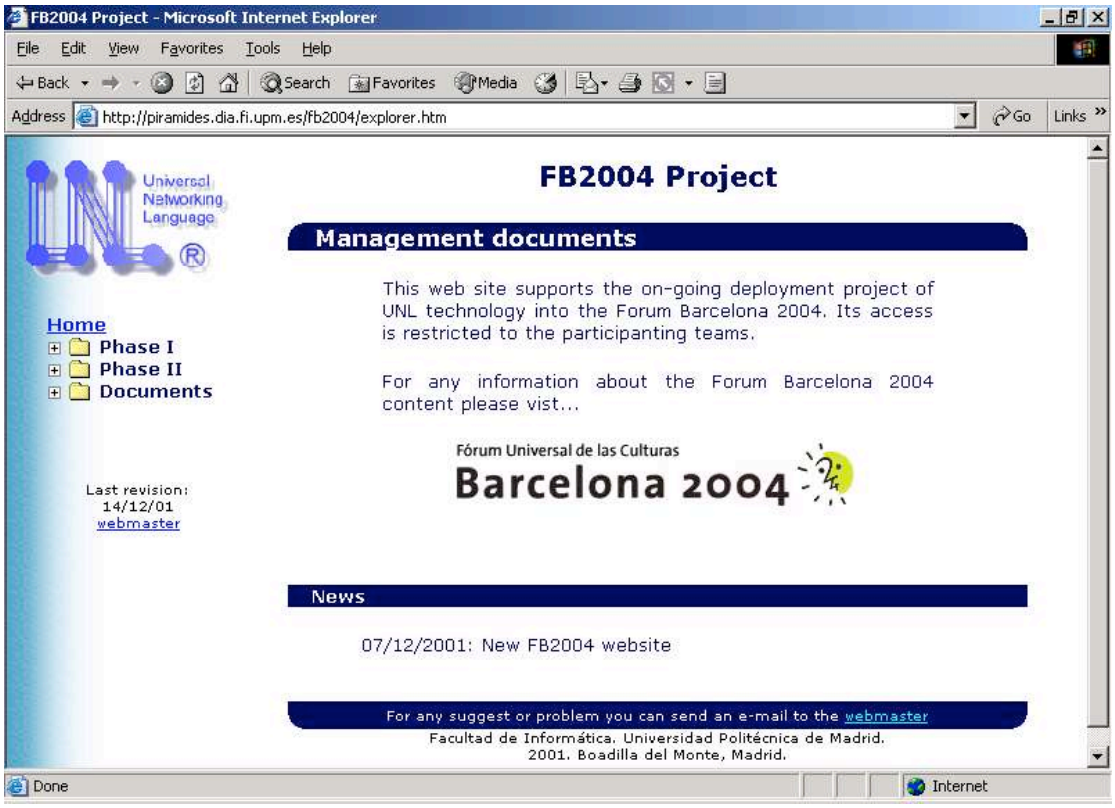


Fig. C-38 Page d'accueil du projet FB2004

Voici extrait de ce corpus XML-isé :

```
<?xml version="1.0" encoding="Unicode"?>
<!--<?xml-stylesheet type="text/xsl" href="unifem.xsl"?> -->
<unl:D unl:dn="FB2004" unl:on="Symposium 2001 Geneva "
unl:dt="2001"
xmlns:unl="http://www.undl.org/2002/schema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.undl.org/2002/schema
UNL-XML.xsd">
<unl:P number="1">
<unl:S number="1">
<unl:org lang="el">
The Universal Forum of Cultures - Barcelona 2004
</unl:org>
<unl:unl>
mod:01(forum.@def.@entry,universal(mod&lt;thing))
mod:01(forum.@def.@entry,culture(icl>abstract thing).@def.@pl)
cnt(:01.@entry.@title,Barcelona_2004)
</unl:unl>
```

```
<unl:GS lang="es">
el foro universal de las culturas, Barcelona_2004.
</unl:GS>
<unl:GS lang="ru">
_____________ _____ _______ - Barcelona 2004. </unl:GS>
<unl:GS lang="it">
Il forum universale delle culture , Barcelona_2004 , .
</unl:GS>
<unl:GS lang="hd">
saMskQwiyoM kI sArvaBOmika saMgoRTI bArsilonA 2004 .
</unl:GS>
<unl:GS lang="fr">
Forum universel des cultures , Barcelone_2004 , .
</unl:GS>
</unl:S>
```

L'importance de ce corpus est que c'est la première fois qu'on pense à se mettre d'accord sur l'encodage en graphes UNL en discutant et modifiant ces graphes (grâce à un forum web) avant de les déconvertir. Ainsi, les graphes UNL sont plus neutres, moins influencés par la langue de l'équipe qui les a produits.

On a aussi défini une procédure pour se mettre d'accord sur les UW employées.

2.1.8 La main à la pâte

La main à la pâte et **Sèvres** sont le résultat d'une coopération entre le GETA et l'association « La main à la pâte ». Cette association maintient un site web qui permet à des enseignants de sept pays d'échanger leurs méthodes et outils pédagogiques pour l'enseignement des sciences dans le primaire.

Voici la page d'accueil de « La main à la pâte ».

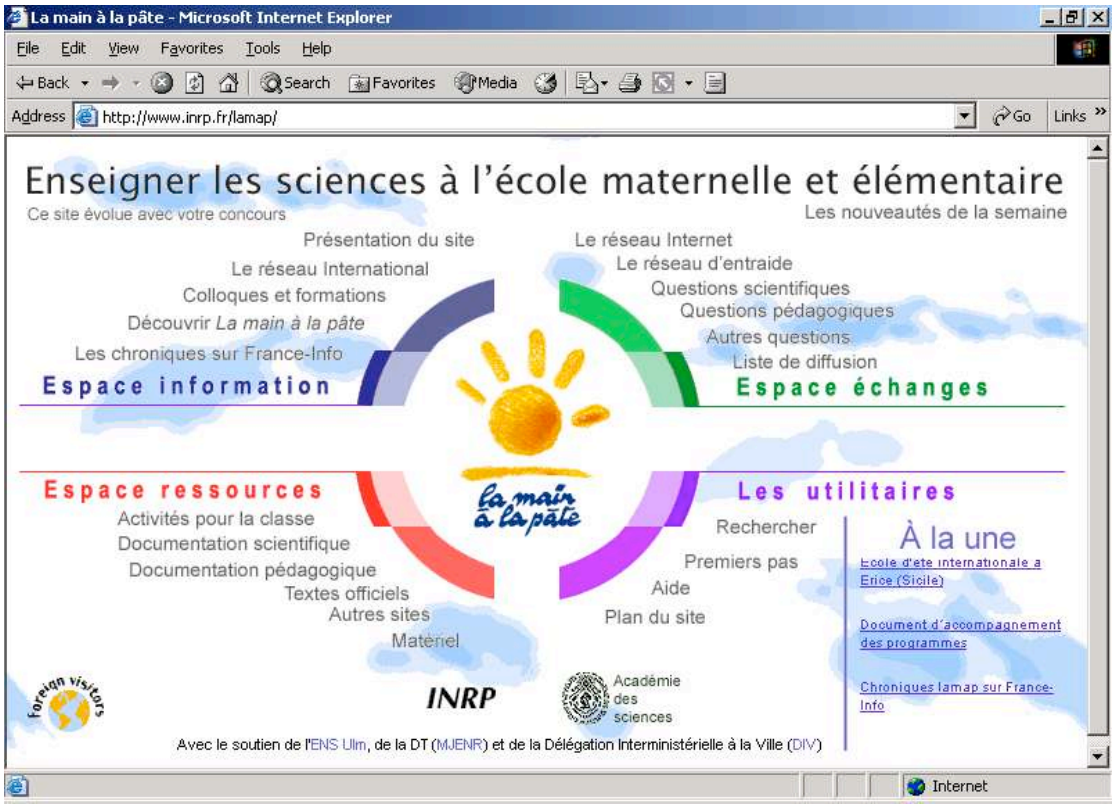


Fig. C-39 Page d'accueil du site « La main à la pâte »

Cette coopération vise à tester la faisabilité de la coédition sur un vrai site web multilingue et avec des utilisateurs ordinaires.

Le texte **Sèvres** est une collection des dix principes du site « La main à la pâte »[15] en six langues ; nous avons ajouté leur graphe UNL aux deux premières phrases. Ce texte a été fait seulement pour la démonstration du premier contact avec les membres de ce projet.

La main à la pâte est le corpus produit dans le cadre du projet, qui est au format UNL-xml étendu. Il contient dix phrases. Chaque phrase a un graphe UNL original construit à partir du texte français, et a été déconverti en français, russe, italien, et espagnol. Après la première déconversion, chaque équipe a modifié son déconvertisseur et/ou son dictionnaire, et parfois complété le graphe UNL, pour obtenir une deuxième déconversion améliorée. Tout cela a été sauvegardé dans le corpus pour permettre de comparer les résultats de l'amélioration. De nouvelles balises ont été ajoutées dans ce corpus pour marquer les versions différentes et le graphe d'où est obtenu chaque résultat de déconversion.

Voici un extrait un peu long qui montre la première phrase de ce corpus. On voit le graphe original et les graphes améliorés par les équipes espagnole et italienne. On voit, par exemple, deux versions en russe ; l'une provient du graphe original, l'autre est la version améliorée après l'ajout de nouvelles UW dans le dictionnaire.

```
<unl:D unl:dn="mainalapate" unl:on="WJT" unl:dt="1/12/2003"
```

[15] La main à la pâte - http://www.inrp.fr/iamap/

```
xmlns:unl="http://www-clips.imag.fr//geta/User/wang-
ju.tsai/detaform/unldoc1.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.undl.org/2002/schema
UNL-XML.xsd">

<!--
toutes les version1 deconverties le 8/7/2003
fr version2: new server, modif on dico, 28/11/2003;
ru version2: different deconverter, modification of dico,
12/09/2003 ;
es version2: modif on dico and graph, 07/09/2003;
it version2: modif on dico, modif on graph to adapt grammar
rule on italian server, 14/09/2003;
-->
<unl:P number="1">
<unl:S number="1">
<unl:org lang="fr">
Ce site est avant tout un outil pour les enseignants du
primaire (élèves âgés de 3 à 11 ans) souhaitant pratiquer les
sciences et la technologie en classe.
</unl:org>
<unl:unl version="fr0">
aoj(tool(icl>thing).@entry,website)
mod(website,this(mod&lt;thing))
man(tool(icl>thing).@entry,first(icl>how))
ben(tool(icl>thing).@entry,teacher.@def.@pl)
agt(wish(icl>do),teacher.@def.@pl)
obj(wish(icl>do),practise(icl>do))
agt(practise(icl>do),teacher.@def.@pl)
obj(practise(icl>do),technology.@def)
and(technology.@def,science(icl>study).@def.@pl)
plc(practise(icl>do),classroom.@def)
mod(teacher.@def.@pl,school.@def.@pl)
mod(school.@def.@pl,elementary(mod&lt;school))
mod(school.@def.@pl,:01.@parenthesis)
aoj:01(range(icl>be).@entry,age(icl>state).@def)
pos:01(age(icl>state).@def,pupil.@def.@pl)
man:01(range(icl>be).@entry,:02)
qua:02(year(icl>period):01.@entry,3)
fmt:02(year(icl>period):01.@entry,year(icl>period):02)
qua:02(year(icl>period):02,11)
</unl:unl>
<unl:unl version="es1">
aoj(tool(icl>thing).@entry, website)
mod(website,this(mod&lt;thing))
man(tool(icl>thing).@entry, first(icl>how))
ben(tool(icl>thing).@entry, teacher.@def.@pl)
agt(wish(icl>do),teacher.@def.@pl)
obj(wish(icl>do),practise(icl>do))
agt(practise(icl>do),teacher.@def.@pl)
obj(practise(icl>do), technology.@def)
and(technology.@def, science(icl>study).@def.@pl)
plc(practise(icl>do), classroom.@def)
mod(teacher.@def.@pl, school.@def.@pl)
mod(school.@def.@pl, elementary(mod&lt;school))
```

```
mod(school.@def.@pl, :01.@parenthesis)
aoj:01(range(icl>be).@entry, age(icl>state).@def)
pos:01(age(icl>state).@def, pupil.@def.@pl)
man:01(range(icl>be).@entry, :02)
qua:02(year(icl>period):01.@entry, 3)
fmt:02(year(icl>period):01.@entry, year(icl>period):02)
qua:02(year(icl>period):02, 11)
</unl:unl>
<unl:unl version="it1">
aoj(tool(icl>thing).@entry.@indef,website)
mod(website,this(mod&lt;thing))
man(tool(icl>thing).@entry,first(icl>how))
ben(tool(icl>thing).@entry,teacher.@def.@pl)
agt(wish(icl>do),teacher.@def.@pl)
obj(wish(icl>do).@present,practise(icl>do))
agt(practise(icl>do),teacher.@def.@pl)
obj(practise(icl>do),technology.@def)
and(technology.@def,science(icl>study).@def.@pl)
plc(practise(icl>do),classroom.@def)
mod(teacher.@def.@pl,school.@def.@pl)
mod(school.@def.@pl,elementary(mod&lt;school))
mod(school.@def.@pl,:01.@parenthesis)
aoj:01(range(icl>be).@entry,age(icl>state).@def)
pos:01(age(icl>state).@def,pupil.@def.@pl)
man:01(range(icl>be).@entry,:02)
qua:02(year(icl>period):01.@entry,3)
fmt:02(year(icl>period):01.@entry,year(icl>period):02)
qua:02(year(icl>period):02,11)
</unl:unl>
<unl:GS lang="el" version="1" graphorg="manuel">This website
is first of all a tool for teachers of elementary school
(pupils' age ranges from 3 to 11 years old) wishing to
practise science and technology in the classroom.
</unl:GS>
<unl:GS lang="es" version="1" graphorg="fr0">website este es
herramienta de forma primera para los maestros de las escuelas
de elementary(mod&lt;school) de que la edad de los alumnos sea
rango tres desde años a once años que desean que practican las
ciencias y la tecnología en el aula.
</unl:GS>
<unl:GS lang="es" version="2" graphorg="es1">este sitio web es
herramienta ante todo para los maestros de las escuelas de
primaria de que la edad de los alumnos oscile tres desde años
a once años que desean que practican las ciencias y la
tecnología en el aula.
</unl:GS>
<unl:GS lang="ru" version="1" graphorg="fr0">___ ________-
________ - _________ _______ ___ ________ _________ ____ (,
_______ ________ ______ _ 3 ___ __ 11 ___), _______ ______
___________ _____ _ _________ _ ________ _______.
</unl:GS>
<unl:GS lang="ru" version="2" graphorg="ru1">___ ________-
________ - _________ ___ ____ ___ ________ _________ ____,
(_______ ________ ______ _ 3 ___ __ 11 ___), _______ ______
___________ _____ _ _________ _ ________ _______.
</unl:GS>
```

```
<unl:GS lang="it" version="1" graphorg="fr0">website e'
strumento per i teacher che esercit classroom la tecnologia e
le scienze che augur che alfabetizzano .
</unl:GS>
<unl:GS lang="it" version="2" graphorg="it1">questo sito e'
uno strumento per i professori delle scuole elementari (l'
eta' degli allievi che var da 3 anno a 11 anno )che pratic
nella classe la tecnologia e le scienze che vogliono praticare
.
</unl:GS>
<unl:GS lang="fr" version="1" graphorg="fr0"> &lt;cette>
&lt;&lt;WEBSITE>> est un outil unième pour les &lt;prof> des
écoles élémentaires ( que vieillit les &lt;pupille> est une
plage ) 3 années &lt;11> années un &lt;voeu> qu'ils se
&lt;exercer_s> les sciences et la technologie dans la
&lt;salle_de_classe>
</unl:GS>
<unl:GS lang="fr" version="2" graphorg="fr0">*CE? &lt;> EST UN
OUTIL UNIE!2ME LES DES E!1COLES E!1LE!1MENTAIRES ( QUE
VIEILLIT LES EST UNE GAMME ) 3 ANNE!1ES &lt;11> ANNE!1ES UN
DE!1SIR QU'ILS SE AUX SCIENCES ET A!2 LA TECHNOLOGIE DANS LA
</unl:GS>
</unl:S>
```

On note des améliorations substantielles des résultats produits par les versions corrigées des déconvertisseurs.

Avec une XSLT, l'utilisateur peut voir en parallèle toutes les phrases et donc comparer le changement de l'amélioration. C'est l'esprit de la coédition. Hajlaoui [Hajlaoui 03] travaille à la réalisation de cet outil.

2.1.9 UNL-HEREIN

UNL-HEREIN est un document du site web « European Heritage »[16] proposé par l'équipe espagnole. L'objectif est de produire des pages multilingues en utilisant UNL. Le corpus est divisé en trois parties qui correspondent aux 3 sections de texte dans la page web. Ces trois parties contiennent séparément 23, 150, et 49 phrases. La seconde contient beaucoup de noms propres, titres, et adresses.

Voici une page sur le site « European Heritage » qui a été encodée en UNL.

[16] Réseau Européen du Patrimoine (European Heritage Netzork) - http://www.european-heritage.net/sdx/herein/index.xsp

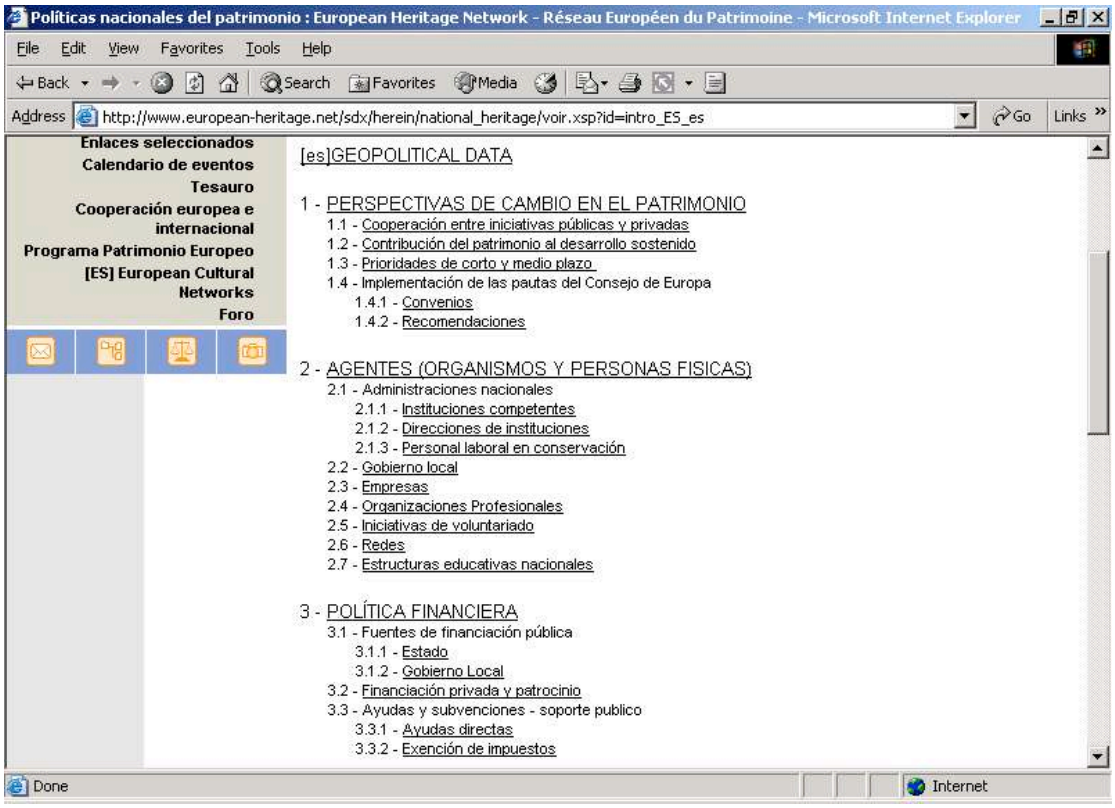


Fig. C-40 page web de « European Heritage » à encoder en UNL

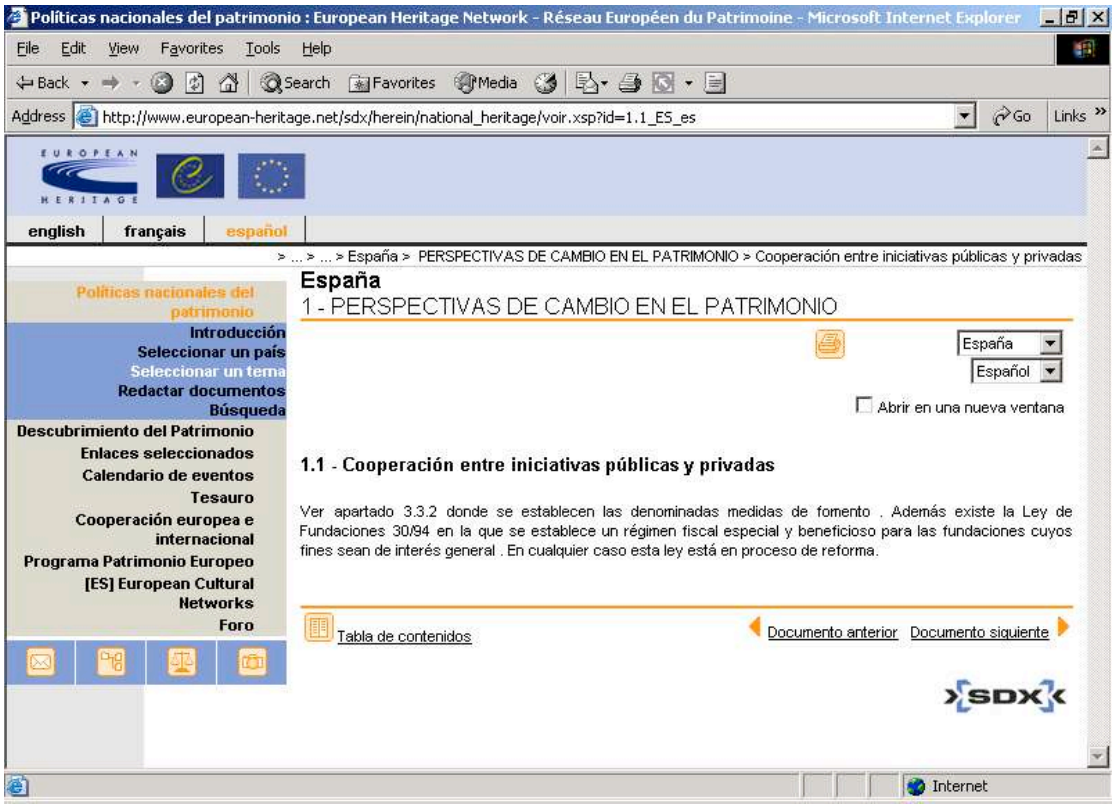


Fig. C-41 Page correspondant à l'extrait du corpus

Voici l'extrait du corpus qui correspond aux phrases montrées dans la Fig. C-41.

```
;NATIONAL POLICIES
;1.  CHANGING PERSPECTIVES ON HERITAGE STRATEGIES
;1.1 Co-operation between private and public initiatives
[S:1]
{org}
Ver apartado 3.3.2 donde se establecen las denominadas medidas
de fomento.
{/org}
{unl}
obj(look(agt>thing,obj>thing).@entry.@obligation,
section(icl>chapter).@def)
nam(section(icl>chapter).@def,"3.3.2")
scn(establish(agt>thing,obj>thing), section(icl>chapter).@def)
obj(establish(agt>thing,obj>thing),
measurement(icl>thing).@pl.@def)
mod(measurement(icl>thing).@pl.@def, promotion(icl>event))
{/unl}
[/S]
[S:2]
{org}
Además existe la Ley de Fundaciones 30/94 en la que se
establece un régimen fiscal especial y beneficioso para las
fundaciones cuyos fines sean de interés general.
{/org}
{unl}
obj(exist(icl>be).@entry, law(icl>rules).@def)
mod(law(icl>rules).@def,foundation(icl>institution):01.@pl)
nam(law(icl>rules).@def, "30/94")
scn(establish(agt>thing,obj>thing),law(icl>rules).@def)
obj(establish(agt>thing,obj>thing),regime(icl>government).@ind
ef)
aoj(regime(icl>government).@indef,beneficial(aoj>thing))
and(beneficial(aoj>thing),special(aoj>thing))
ben(beneficial(aoj>thing),foundation(icl>institution):02.@pl.@
def)
pos(objective.@pl,foundation(icl>institution):02.@pl.@def)
aoj(have(aoj>thing,obj>thing),objective.@pl)
obj(have(aoj>thing,obj>thing),interest(icl>thing).@indef)
mod(interest(icl>thing).@indef, general(aoj>thing))
{/unl}
[/S]
```

Le projet HEREIN-UNL est terminé, et seulement un corpus au format UNL-xml a été produit. Les phrases multilingues n'ont pas été adaptées sur le site HEREIN parce que le site lui-même n'est pas prêt à intégrer des applications UNL et XML.

Tous ces corpus sont sous forme textuelle, avec des codages différents. Nous en avons choisi plusieurs qui sont fortement multilingues pour les transformer en XML. Ils sont disponibles sur le site web SWIIVRE-UNL. Nous les avons utilisés pour notre étude expérimentale des correspondances. Dans l'avenir, nous comptons déconvertir toutes les versions qui manquent et XML-iser tous les corpus.

2.2 Hiérarchie dans la modélisation d'une correspondance graphe-texte

La première étape de cette étude nous a conduit à introduire une certaine hiérarchie entre les correspondances observées. En effet, quand on observe les graphes UNL et les textes correspondants dans ces corpus, on constate qu'il y a des niveaux de granularité différents.

Pour étudier cela de plus près, nous avons fait passer un analyseur morpho-syntaxique (AMS) ou un segmenteur, selon la langue, et nous avons étudié les correspondances entre graphes et treillis LMS.

2.2.1 Côté texte: phrase ⊃ mot ⊃ lemme/affixe ⊃ information grammaticale

Nous n'entrons pas dans les détails de la discussion des définitions de ce que sont les mots, ou les parties des discours (POS). Puisque nous utilisons des ressources existantes, nous définissons cette hiérarchie selon les sorties de ces ressources.

Une phrase est un énoncé d'une langue. La plupart de temps, c'est l'unité d'entrée d'un analyseur morpho-syntaxique (AMS) ou d'un segmenteur.

Une phrase est segmentée en « mots » par le processus d'AMS (ou de simple segmentation), éventuellement de plusieurs façons concurrentes (d'où une représentation du résultat par un treillis). Dans la pratique, les « mots » ainsi segmentés sont des formes (fléchies ou non) de lemmes (simples ou composés).

Les informations grammaticales correspondent aux catégories grammaticales et autres variables grammaticales (genre, nombre, personne, temps, mode..).

2.2.2 Côté graphe: graphe/sous-graphe/scope ⊃ arc ⊃ nœud/relation ⊃ UW/restriction/ attribut

Nous distinguons quatre niveaux hiérarchiques dans un graphe UNL :

> Graphe, sous-graphe et scope : un sous-graphe est défini par un sous-ensemble des arcs d'un graphe. (La définition formelle du « scope » est donnée en section B.2.3.1.)
>
> Arc : un graphe UNL se compose d'arcs et de nœuds ; un arc relie deux nœuds (chacun pouvant être un « scope », ou sous-graphe « replié ») et porte une relation sémantique.
>
> Nœud : dans un nœud, nous pouvons encore distinguer l'UW, la liste de restrictions, et les attributs.
>
> UW.

2.2.3 Les correspondances identifiées

Les correspondances identifiées ici sont tirées des corpus UNL. Ainsi, nous évitons des exemples atypiques ou rares.

Voici un tableau des correspondances que nous avons identifiées, marquées par un « X ».

Graphe / Texte	Graphe	sous-graphe/ scope	arc	relation	nœud	UW	restriction	attribut
Phrase	X							
Sous-chaîne		X	X		X			
Mot	X	X	X	X	X	X	X	X
Morphème /lemme/affixe				X	X	X		
Info grammaticale				X			X	X

Tableau C-2 Types de correspondance entre graphe UNL et LN

Nous continuons en donnant quelques exemples pour chaque type de correspondance.

2.3 Correspondances lexicales

Commençons par les correspondances lexicales. La plupart des exemples qui suivent viennent de nos corpus et quelques-uns ont été ajoutés pour compléter, en fonction de nos connaissances d'UNL et d'autres langues naturelles.

Il y a plusieurs types de correspondance lexicale. Pour chacun d'eux, nous donnons un ou plusieurs exemples. Chaque exemple comprend une partie de graphe UNL et le texte correspondant. Les parties correspondantes sont soulignées. Les textes sont tous au moins dans la langue d'origine.

Ici, la correspondance lexicale est obtenue en observant simplement le texte et le graphe. L'unité du texte étant le mot, on part des mots et on observe les parties correspondantes dans le graphe. La correspondance lexicale est donc la correspondance entre la forme de surface du texte et le graphe.

2.3.1 Graphe / mot

exemple (I)

{unl} agt(run.@entry.@request, you) {/unl}

cours! (français)
run! (anglais)

2.3.2 Arc / mot

exemple (I)

agt(drink.@entry.@past, you)

bebiste (espagnol)

exemple (II)

man(explain(agt>thing, obj>thing), in detail(icl>how))

détailler (français)

2.3.3 Relation / mot

exemple (I)

plc(resolution(icl>decision).@entry.@def, Suzhou(icl>city))

_____ (japonais)
_____ (chinois)
_____ _ _____ (russe)

exemple (II)

aoj(tool(icl>thing).@entry,website)
mod(website,this(mod<thing))

Ce site est un outil. (français)

exemple (III)

pos : _ (chinois) _(japonais) ___ (thaï)
plf : _ (chinois) __(japonais) c (russe)
pur : ___ (russe), pour (français)

2.3.4 Nœud + relation / mot

exemple (I)

pos(*, he)

his (anglais)
su (espagnol)

2.3.5 Nœud / mot

exemple (I)

agt(provide(icl>give).@entry.@future, UNDL
Foundation(icl>foundation).@def)

La fondation UNDL fournira (français)
UNDL Foundation proporcionará (espagnol)
La Fondazione UNDL fornirà (italien)

exemple (II)

drink(icl>do).@past : drank (anglais)
you.@polite: vous (français)
man(icl>human).@pl : men (anglais) _ (chinois)

2.3.6 UW / mot

exemple (I)

plc:03(resolution(icl>decision).@entry.@def, Suzhou(icl>city))

_____(japonais)
_____ (chinois)
_____ _ _____ (russe)

exemple (II)

marry(icl>do, agt>male) : _ (chinois) _____ (russe)
marry(icl>do, agt>female) : _ (chinois) _____ (russe)

2.3.7 Restriction / mot

(icl>human, sex>male) : _ (chinois, japonais) _____ (thaï)
(icl>human, sex>female) : _ (chinois, japonais) _____ (thaï)
(icl>animal, sex>male) : _ (chinois) _____ (thaï)
(icl>animal, sex>female) : _ (chinois) _____ (thaï)

2.3.8 Attribut / mot

exemple (I)

agt(provide(icl>give).@entry.@future, UNDL
Foundation(icl>foundation).@def)

UNDL ____ (chinois)
UNLD Foundation will provide (anglais)

exemple (II)

or(switch off(icl>do).@entry.@possibility,switch on(icl>do).@possibility)

_____ ____ _____ ___ _____. (russe)

exemple (III)

.@complete : _ (chinois), ____ (thaï), _ (japonais)
.@not: _ (chinois), not (anglais), ___ (thaï)

2.4 Correspondances d'attributs

La correspondance d'attributs ne peut pas être obtenue en observant le texte et le graphe, elle vient de la structure plus profonde des mots produite par l'AMS. Il faut donc considérer le treillis LMS, où chaque mot porte des informations grammaticales et son lemme. Pour trouver une correspondance d'attributs, on part des attributs des mots et on cherche la partie correspondante du graphe.

La liste d'attributs d'un mot est de la forme suivante : (lemme, catégorie grammaticale, autres informations grammaticales). Les parties correspondantes dans le graphe et dans la liste d'attributs sont soulignées.

2.4.1 Headword, UW, nœud / lemme

exemple (I)

eye.@pl.@def : les yeux
_(œil, nom, pluriel)

exemple (II)

nam(sea.@def, Aral)
plc(live(icl>do).@past.@entry, sea.@def)
agt(live(icl>do).@past.@entry, species.@pl)

En el mar Aral vivían
_(vivir, verbe, 3ème personne, pluriel, imparfait)

exemple (III)

mod(function(icl>abstract thing).@entry.@pl, basic(mod<thing))
mod(function(icl>abstract thing).@entry.@pl, tuner.@def)

_____ _____ _____
_(_____, nom, pluriel, nominatif)

2.4.2 Relation / lemme

aoj(meter(icl>unit).@pl.@past.@entry, deepness)
qua(meter(icl>unit).@pl.@past.@entry, 16)

mod(deepness, average(mod<thing)) pos(deepness, it)

Su profundidad media era de 16 metros (espagnol)
_(ser , verbe, 3ème personne, singulier, prétérite)

2.4.3 Relation / affixe

mod(resource(icl>abstract thing).@pl, natural(aoj>thing))
mod(resource(icl>abstract thing).@pl, Africa(icl>region))

Africa's natural resources.

2.4.4 Relation / information grammaticale

mod(function(icl>abstract thing).@entry.@pl, basic(mod<thing))
mod(function(icl>abstract thing).@entry.@pl, tuner.@def)

_____ _____ _____
_(_____, nom, singulier, génitif)

2.4.5 Restriction / information grammaticale

exemple (I)

mod(function(icl>abstract thing).@entry.@pl, basic(mod<thing))
mod(function(icl>abstract thing).@entry.@pl, tuner.@def)

_____ _____ _____
_(_____, nom, pluriel, nominatif)

exemple (II)

nam(sea.@def, Aral)
plc(live(icl>do).@past.@entry, sea.@def)
agt(live(icl>do).@past.@entry, species.@pl)

En el mar Aral vivían
_(vivir, verbe, 3eme personne, pluriel, imparfait)

exemple (III)

mod(function(icl>abstract thing).@entry.@pl, basic(mod<thing))
mod(function(icl>abstract thing).@entry.@pl, tuner.@def)

_____ _____ _____
_(_____, adjectif, pluriel, nominatif)

2.4.6 Attribut / information grammaticale

mod(function(icl>abstract thing).@entry.@pl, basic(mod<thing))
mod(function(icl>abstract thing).@entry.@pl, tuner.@def)

_____ _____ _____
_(_____, nom, pluriel, nominatif)

2.5 Correspondances structurales

Les correspondances structurales sont les correspondances entre une partie du graphe et une partie du texte.

2.5.1 Graphe entier / phrase entière

Selon la conception d'UNL, une phrase en langue naturelle correspond à un graphe UNL.

{unl}
obj(devote(icl>do).@entry.@future, month(icl>date).@def.@topic)
nam(month(icl>date).@def.@topic, April) gol(devote(icl>do).@entry.@future, :01)
and:01(poetry(icl>art(icl>abstract thing)).@entry.@generic,
iterature(icl>art(icl>abstract thing)).@generic)
{/unl}

Le mois d'avril sera consacré à la littérature et à la poésie.

2.5.2 Sous-graphe quelconque / sous-chaîne

Ici, le sous-graphe peut être connexe ou non, si ce sous-graphe contient des arcs de deux scopes différents, il est souvent non connexe, et la sous-chaîne peut être connexe ou non. L'exemple suivant montre la correspondance entre un sous-graphe non connexe et une sous-chaîne non connexe.

{unl}
obj(devote(icl>do).@entry.@future, month(icl>date).@def.@topic)
nam(month(icl>date).@def.@topic, April) gol(devote(icl>do).@entry.@future, :01)
and:01(poetry(icl>art(icl>abstract thing)).@entry.@generic,
iterature(icl>art(icl>abstract thing)).@generic)
{/unl}

Le mois d'avril sera consacré à la littérature et à la poésie.

2.5.3 Scope / sous-chaîne

{unl}
obj(devote(icl>do).@entry.@future, month(icl>date).@def.@topic)
nam(month(icl>date).@def.@topic, April) gol(devote(icl>do).@entry.@future, :01)
and:01(poetry(icl>art(icl>abstract thing)).@entry.@generic,
iterature(icl>art(icl>abstract thing)).@generic)
{/unl}

Le mois d'avril sera consacré à la littérature et à la poésie.

2.5.4 Arc / sous-chaîne

Enfin, un arc dans un graphe UNL correspond très souvent à une paire prédicat-argument ou modifiant-modifié.

obj:01(on(icl>about), build(agt>thing,obj>thing))
obj:01(build(agt>thing,obj>thing), knowledge(icl>information))
mod:01(knowledge(icl>information), global(icl>worldwide))

on Building Global Knowledge (anglais)
sur la construction de connaissance globale (français)
_____ (japonais)
_____ _____ (russe)

2.6 Remarques sur les correspondances

Nous pouvons constater qu'il y a des correspondances très évidentes, faciles à trouver, par exemple, la correspondance « mot vedette – lemme » : si on peut trouver la paire « mot vedette – lemme » dans un dictionnaire anglais-L, on est presque sûr que cela est une correspondance identifiée.

Puis il y des attributs ou des restrictions qui peuvent aider à créer des liens vers les informations grammaticales. Par exemple, pour les langues examinées, un mot vedette suivi par la restriction (icl>do) correspond forcément à un verbe ou à un substantif d'action, (icl>thing) à un nom, (icl>how) à un adverbe ou à une périphrase adverbiale (ex : urgently(icl>how) _ de façon urgente), et (mod<thing) à un adjectif ou à une qualificative.

Il n'est pas surprenant que certains types de correspondances soient plus évidents dans certaines langues que dans d'autres. Par exemple, dans des langues comme le chinois qui ne distingue pas singulier/pluriel et dont les catégories grammaticales sont assez vagues, ces correspondances sont moins évidentes que celles du russe ou du français.

Un autre point important est qu'il y des schémas de correspondance qui ne se produisent pas tout le temps. Par exemple, les particules du futur en chinois « _ » et en thaï « __ » sont très souvent facultatives, et donc l'apparition de .@future dans un graphe ne garantit pas l'apparition de ces particules.

En bref, les types de correspondance et leur degré de régularité varient selon la langue naturelle en question. Nous dirons qu'une correspondance est « forte » si elle est très régulière et elle sera donc a priori appliquée pendant la procédure de construction de liens, par exemple, .@pl_pluriel , (icl>thing)_substantif, etc.

Enfin, cette analyse n'est qu'un premier essai, et il est possible que d'autres types de correspondances UNL-LN apparaissent dans le futur. En particulier, nous n'avons pu étudier aucun exemple de dialogue, et on peut penser qu'on y trouverait des correspondances entre attributs pragmatiques dans le graphe et expressions idiomatiques ou certaines combinaisons de valeurs d'attributs (conditionnel pour la politesse, par exemple).

3. Formalisation et calcul possible des correspondances graphe-texte

Nous avons identifié un certain nombre de types de correspondance entre les graphes UNL et les textes. Nous allons maintenant les utiliser pour construire des liens, une fois que le texte et le graphe auront été légèrement traités, i.e. transformés respectivement en un treillis LMS et un arbre UNL.

Pour cela, il nous faut d'abord formaliser les correspondances sous forme de structures implémentables, puis construire un algorithme de calcul des correspondances.

Notre formalisation consiste à représenter les correspondances par des « liaisons » entre éléments de deux structures. Par exemple, une liaison texte-treillis sera de forme (sous-chaîne, nœud), et une liaison arbre-treillis de forme (liste de nœuds, liste de nœuds).

3.1 Contraintes sur la représentation et le calcul des correspondances

Remarquons d'abord que nous ne pouvons pas réutiliser les représentations de correspondance chaîne-arbre présentées plus haut, car aucune n'est assez détaillée.

Par exemple, la représentation des SSTC par SNODE et STREE ne suffit pas, parce qu'un nœud dans le graphe UNL contient beaucoup d'information, comme un nœud dans un arbre de m-structure du GETA. Il nous faut pouvoir marquer les correspondances entre attributs et informations grammaticales.

D'autre part, les arcs dans le graphe UNL, avec les relations sémantiques associées, nous font penser aux règles sagittales de la grammaire transductive ou à la base de données de patrons de correspondances du système PIVOT que nous avons discuté en partie B section 1.2.4. Mais les règles sagittales de la grammaire transductive ont besoin d'une représentation syntaxique, ce que nous n'aurons pas. Nous ne pouvons pas non plus construire à partir de zéro des bases de données pour enregistrer des correspondances et extraire des règles.

Il nous faudra donc des liaisons entre des éléments complexes (nœud, arc, lemme décoré) aussi bien qu'entre leurs composants aux différents niveaux hiérarchiques vus plus hauts. Reste à imaginer comment calculer ces correspondances.

Ce dont nous avons besoin, c'est d'un algorithme heuristique qui nous permet de créer autant de liens que possible entre le texte et le graphe, puis de choisir une « meilleure correspondance ». Il commencera par construire des liens nœud (d'arbre) – lemme (du

treillis), puis à partir de ces liens nœud-lemme, nous chercherons à appliquer des patrons de correspondances, selon leur sûreté.

Un tel algorithme de « meilleur d'abord » (« best-first ») avec application des règles par sûreté décroissante, est aussi employé par Richardson et Menezes à Microsoft [Menezes 01] et par Watanabe à IBM-Japon [Watanabe 00] pour trouver les meilleures correspondances entre deux arbres ou deux corpus alignés.

Nous présentons ensuite notre algorithme de construction des correspondances. Pour des raisons de simplicité et de clarté de l'exposé et de l'implémentation, nous nous donnons les contraintes suivantes :

> la langue naturelle depuis laquelle nous construisons les correspondances est le français.
>
> l'AMS est PILAF et l'arbre UNL est celui défini au GETA.

L'extension aux autres langues naturelles et à d'autres structures arborescentes n'est pas difficile en théorie, en utilisant le même module général.

3.2 Correspondance entre texte et treillis LMS

Nous décomposons la correspondance texte-graphe UNL en quatre couches de structures et trois correspondances entre ces quatre couches. Pour expliquer la correspondance entre deux couches de structures, nous donnons au début une définition formelle et une formalisation associée. Nous donnons ensuite une description de l'algorithme illustrée par des exemples. Enfin nous spécifions la structure de données et le calcul possible pour implémenter l'algorithme.

3.2.1 Notions de base

Voici d'abord quelques notions de base qui seront utilisées dans notre algorithme et dans notre définition formelle.

Notre structure de données a quatre couches principales :

> couche 1 - texte (S_1)
>
> couche 2 - treillis LMS (S_2)
>
> couche 3 - arbre UNL (S_3)
>
> couche 4 -graphe UNL (S_4)

Une *liaison* l_{ij} est un lien créé entre deux couches de structures (S_i et S_j).

Chaque liaison l_{ij} est un quadruplet l_{ij} =(identificateur de liaison, type de profil, élément(s) de la couche haute, élément(s) de la couche basse).

L_{ij} est l'ensemble de toutes les liaisons qu'on peut construire entre S_i et S_j .

$L_{ij} = \{l_{ij}*\}$.

Une *correspondance* C est un ensemble de liaisons vérifiant une certaine propriété.

Donc une C_{ij} est un sous-ensemble de L_{ij} ($C_{ij} \subseteq L_{ij}$).

Tableau C-3 Notions de base pour les correspondances texte-graphe UNL

Nous détaillons et donnons l'algorithme d'établissement des correspondances entre deux couches successives. Nous expliquons l'algorithme avec deux exemples. Ces deux exemples ont été choisis car ils contiennent les deux cas difficiles dans la construction de la correspondance treillis-arbre UNL.

La première difficulté est qu'il peut exister des circuits et des scopes dans un graphe UNL.

La deuxième est qu'un graphe peut contenir des UW qui se répètent sur des nœuds différents. Cela crée normalement des répétitions de mots dans le texte, et il est difficile de décider quel mot (entre ces mots répétés) correspond à quel nœud.

Le graphe UNL et la phrase française de l'exemple (I) sont :
[S:1]
{unl}
agt(regret(icl>do).@entry, he(icl>human))
obj(regret(icl>do).@entry, :01)
agt:01(come.@entry.@future.@not, you)
and(regret(icl>do).@entry, know(agt>human, obj>event))
agt(know(agt>human, obj>event), he(icl>human))
obj(know(agt>human, obj>event), :01)
{/unl}
{fr}il sait que tu ne viendras pas et il le regrette.{/fr}
[/S]

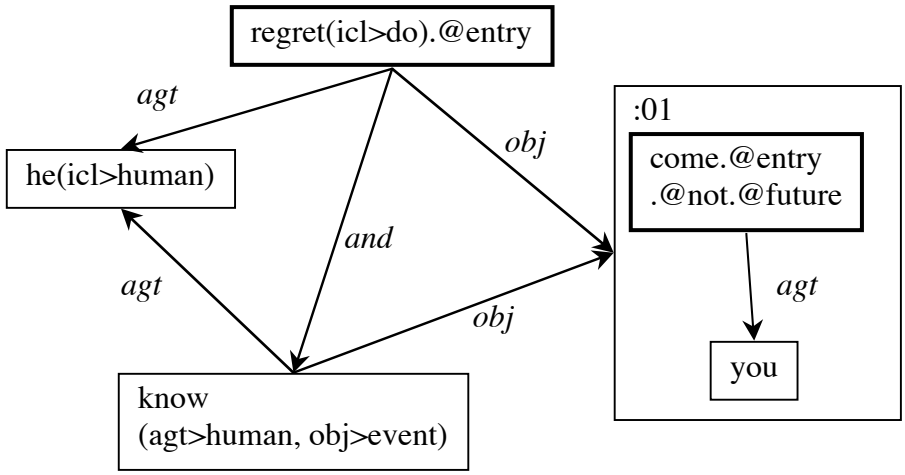


Fig. C-42 Graphe UNL de l'exemple (I)

Le graphe UNL et la phrase française de l'exemple (II) sont :
[S:2]
{unl}
nam(sea:01.@def, Aral)
aoj(sea:02.@def.@entry.@past, sea:01.@def)
mod(sea:02.@def.@entry.@past, inland(mod<thing))
mod(sea:02.@def.@entry.@past, fourth(mod<thing))
mod(sea:02.@def.@entry.@past, large)
man(large, most)

scn(large, world.@def)
{/unl}
{fr}la mer d'Aral était la quatrième plus grande mer intérieure dans le monde{/fr}
[/S]

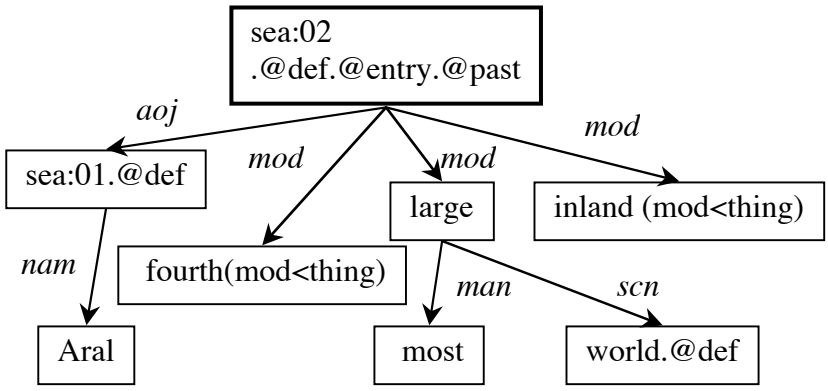


Fig. C-43 Graphe UNL de l'exemple (II) avec deux nœuds « sea »

3.2.2 Définition formelle et formalisation possible

Le calcul de la correspondance entre le texte et le treillis LMS est simple, parce que dans chaque nœud de treillis on trouve directement le lemme et la forme correspondants. Au moment où on construit le treillis LMS, on a toutes les liaisons possibles. Il suffit donc de créer une liste pour garder toutes les liaisons. En utilisant l'AMS PILAF, il n'y a donc qu'un seul profil de liaison entre ces deux couches de structures : mot(s)-nœud.

En fait, il s'agit du côté texte simplement d'une sous-chaîne découpée par PILAF (comme « de plus en plus »). Nous utilisons le terme de « mot » ou « mots », mais PILAF pourrait très bien découper deux lemmes dans un « mot » comme « superautonome » ou « hypermotivé ». Ici, on n'a besoin de définir ce que sont les « mots » que pour l'interface d'édition de façon à définir la sélection par double clic. Il s'agit donc de mots « typographiques » et non linguistiques.

Pour construire les liaisons L_{12}

1) on trouve toutes les liaisons possibles au moment de la lecture du résultat de l'AMS,

2) on dira qu'il y a autant de « correspondances » que de trajectoires dans S_1. ,

Nous avons donc les définitions formelles suivantes :

Une liaison l_{12} = (identificateur, mot(s) de texte, nœud de treillis)
Une correspondance C_{12} est un ensemble de liaisons qui peut être ordonné de façon que la liste de nœuds obtenue soit une sous-trajectoire du treillis. On a donc :
$C_{12} \subseteq L_{12}$.
Correspondance partielle : Nœuds-Treillis (C_{12}) $\subseteq$ Nœuds (t) où t$\in$Trajectoires(S_1)
Correspondance totale : Nœuds-Treillis (C_{12}) = Nœuds (t) où t$\in$Trajectoires(S_1)

Tableau C-4 Définitions formelles pour les correspondances texte-treillis

3.2.3 Structure de données et calcul possible

Nous prenons une phrase comme unité d'entrée du traitement. Une phrase P est simplement une chaîne de caractères.

Une phrase sera segmentée par l'AMS en mots ou expressions, éventuellement avec des chevauchements (par exemple : il le voit peu à peu près tous les jours). Dans le cas de l'AMS PILAF, il n'y a pas de chevauchement[17].

La sélection d'un « mot » à partir d'une position dans le texte peut se faire de deux façons différentes :

- on l'étend à droite et à gauche jusqu'à rencontrer un séparateur, ce que donne un « mot typographique »,
- si le texte est relié à un treillis AMS, on produit une « multisélection » (un peu comme dans KanjiTalk) qui donne l'ensemble des segments découpés par l'AMS et contenant cette position, et on obtient un « mot linguistique ».

Le texte (ici la phrase) sera soumis à PILAF. Le résultat sera alors transformé en un treillis LMS, et en un ensemble de liaisons entre ce treillis et le texte. Chaque nœud de treillis contient un lemme, une forme (le « mot linguistique »), une catégorie, et les valeurs de certaines autres variables.

[17] Par exemple, dans la phrase « Je le vois peu à peu près tous les jours », PILAF trouvera « peu à peu » et « à peu près » ne sera pas trouvé. Avec l'AMS du français écrite en ATEF, l'arbre produit en sortie contient ces deux « formes figées ».

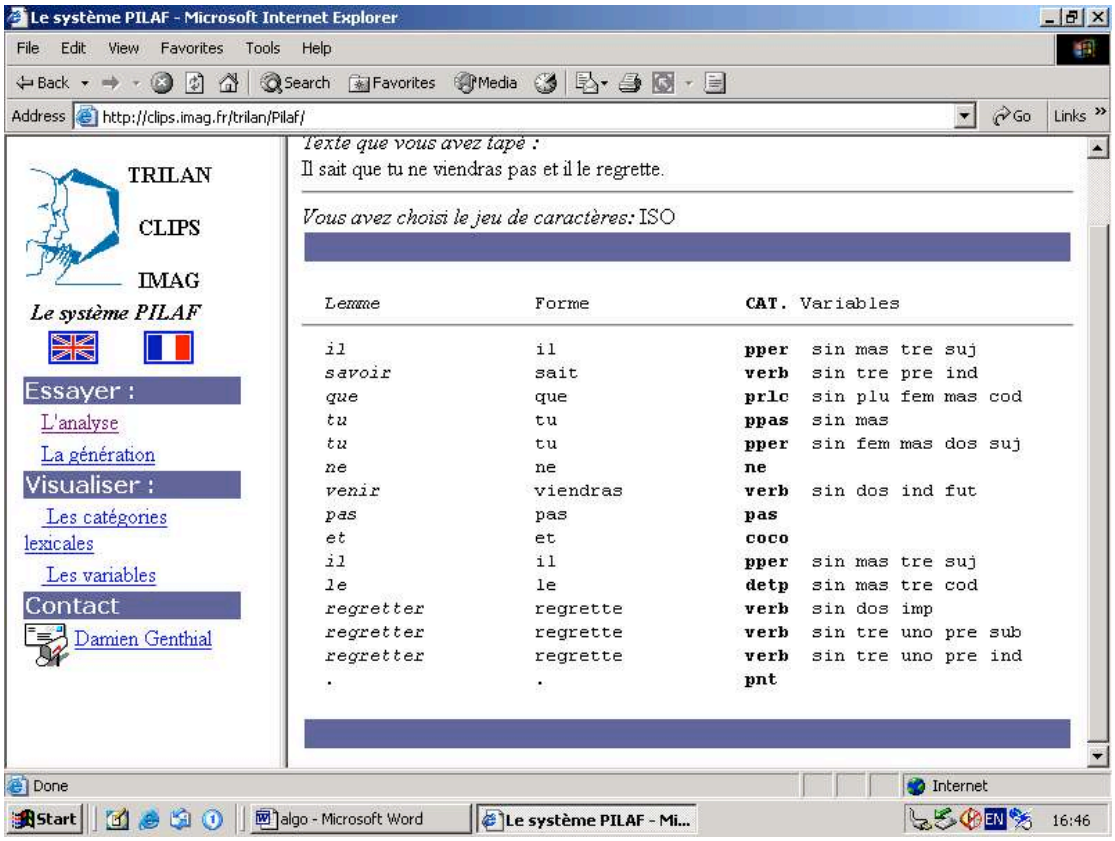


Fig. C-44 Sortie de PILAF de l'exemple (I)

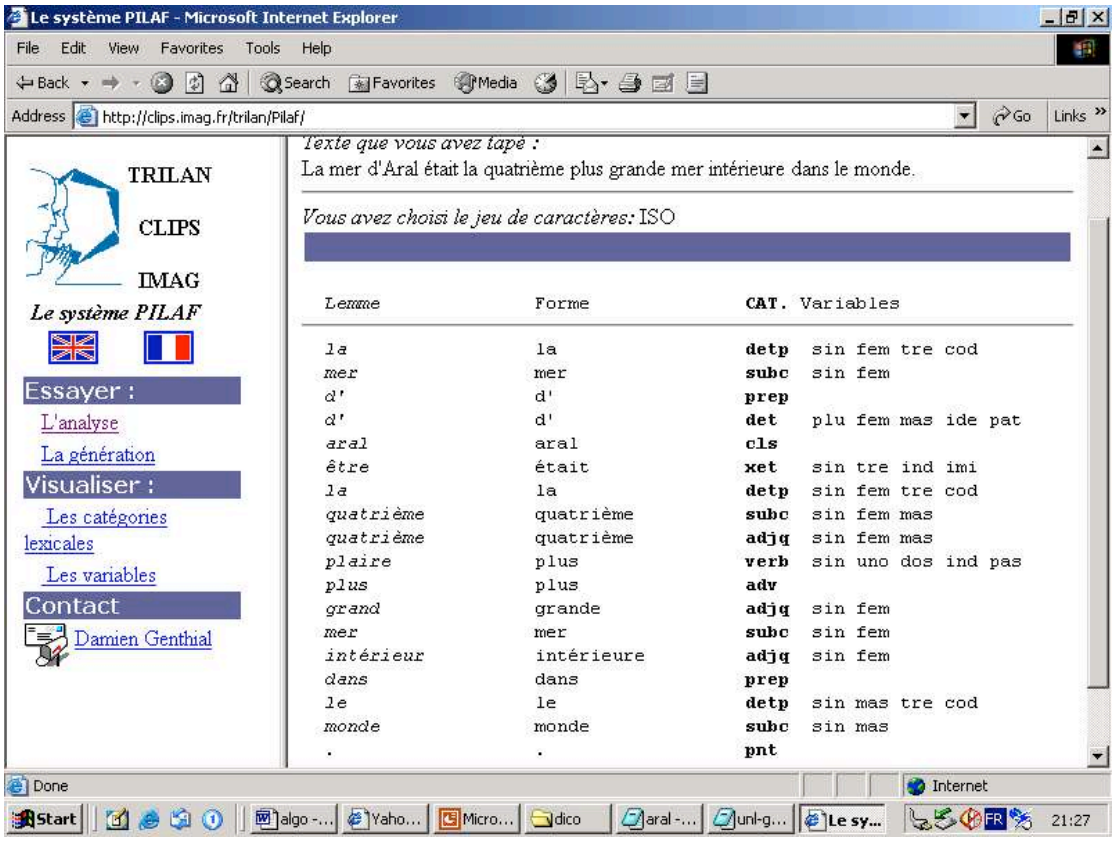


Fig. C-45 Sortie de PILAF de l'exemple (II)

Un nœud de ce treillis (noeudtreillis$_i$) est implémenté par : (id_noeudtr, lemtr, forme, cat, var*, l_precedents, l_suivants). « l_precedents » et « l_suivants » sont deux listes contenant les nœuds précédant et suivant immédiatement ce nœud.

Il y a deux nœuds spéciaux : « début » et « fin », contenant une information linguistique vide. On a donc :

noeudtreillis$_i$ = (id_noeudtr$_i$, lemtr$_i$, forme$_i$, cat$_i$, var$_i$* , l_precedents$_i$, l_suivants$_i$).
T_0 = { débuttreillis, noeudtreillis$_1$, noeudtreillis$_2$, ….noeudtreillis$_n$, fintreillis }

Dans notre exemple (II), on a:
T_0 = {(débuttreillis), (1, la,la, detp, sin/fem/tre/cod, noeuddébut, 2), (2, mer, mer, subc, sin/fem, 1, 3/4) , (3, d', d', prep, - , 2, 5), (4, d', d', det, plu/fem/mas/ide/pat, 2, 5), (5; aral, aral, cls, -, 3/ 4, 6), …(17, monde, monde, subc, sin/mas, 16, noeudfin), (fintreillis)}

Pour chaque nœud de ce treillis, nous consultons ensuite un dictionnaire français-UNL ou français-anglais, et ajouterons au nœud les lemmes anglais (avec éventuellement leurs catégories grammaticales) ou les UW correspondantes.

Ici, nous accédons au dictionnaire français-UNL avec (lemme, catégorie) et ne retenons pas les UW dont la ou les restrictions ne sont pas compatibles avec la catégorie (PILAF) cat. Voici un extrait de la table de compatibilité utilisée.

PILAF		**UW**
catégories		*restriction*
adv	Adverbe	(icl>how)
subc	substantif commun	(icl>thing)
adjq	Adjectif qualificatif	(mod<thing)/(aoj>thing)
verb	Verbe	(icl>do)/(icl>occur)/(icl>state)

Tableau C-5 Table de compatibilité pour treillis étendu

Pour la facilité de l'implémentation de ces variables grammaticales et de ces catégories, nous donnons à chaque nœud une table de catégories et une table de variables. Ces deux tables se composent de variables booléennes. PILAF a au total 42 catégories, 18 variables morphologiques et 5 variables syntaxiques. Les longueurs de ces deux tables sont donc 42 et 23.

Les deux tables du nœud « savoir » dans la Fig. C-46 sont données ci-dessous :

numéro	0	1		10		41	
tab_catégorie	adv	subc	……	verb	……	cls	
valeur Boolean	0	0	……	1	……	0	*longueur=42*

numéro	0	1	2		6		9		14		22	
tab_variable	fem	mas	ind	…	pre	…	sin	…	tre	...	cdn	
valeur Boolean	0	0	1	…	1	…	1	…	1	...	0	*longueur=23*

Ainsi, un nœud étendu (noeudtreillisetendu) se compose de : (id_noeudtr, lemtr, forme, tab_cat, tab_var, (lemmeanglais, catanglais)*, l_precedents, l_suivants).

Nous aurons le treillis étendu T_1 :

$noeudtreillisetendu_i$ = ($id\_noeudtr_i$, $lemtr_i$, $forme_i$, tab_cat_i , tab_var_i , (lemmeanglais, catanglais)*, $l\_precedents_i$, $l\_suivants_i$).
T_1 = { débuttreillis, $noeudtreillisetendu_1$, $noeudtreillisetendu_2$, …, $noeudtreillisetendu_n$, fintreillis }

Dans notre exemple (I), on a[18]:

T_1 = {(débuttreillis), (1, il, il, pper, sin/mas/tre/suj, it/he, débuttreillis, 2), (2, savoir, sait, verb, sin/tre/pre/ind, know, 1, 3/4), (3, que, que, prlc,sin/plu/fem/mas/cod, that, 2, 4/5), (4, tu, tu, pper, sin/fem/mas/dos/suj, you, 3, 6), (5; tu, tu, ppas, sin/mas, quiet, 3, 6), …(12, regretter, regrette, verb, sin/tre/uno/pre/ind, regret, 11, fintreillis), (13, regretter, regrette, verb, sin/tre/uno/pre/sub, regret, 11, fintreillis), (14, regretter, regrette, verb, sin/dos/imp, regret, 11, fintreillis), (fintreillis)}

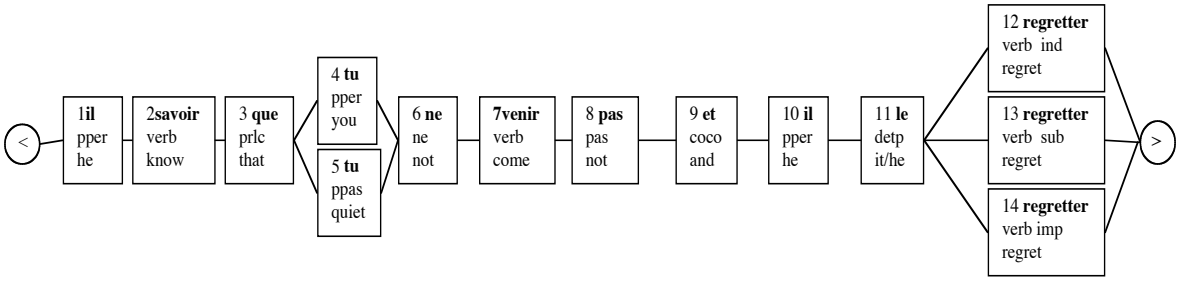


Fig. C-46 Treillis étendu exemple (I)

Dans notre exemple (II), on a :
T_1 = {(débuttreillis), (1, la, la, detp, sin/fem/tre/cod, the, débuttreillis, 2), (2, mer, mer, subc, sin/fem, sea, 1, 3/4), (3, d', d', prep, - , -, 2, 5), (4, d', d', det, plu/fem/mas/ide/pat, some, 2, 5), (5; aral, aral, cls, -, Aral, 3/ 4, 6), …(17, monde, monde, subc, sin/mas, world, 16, fintreillis), (fintreillis)}

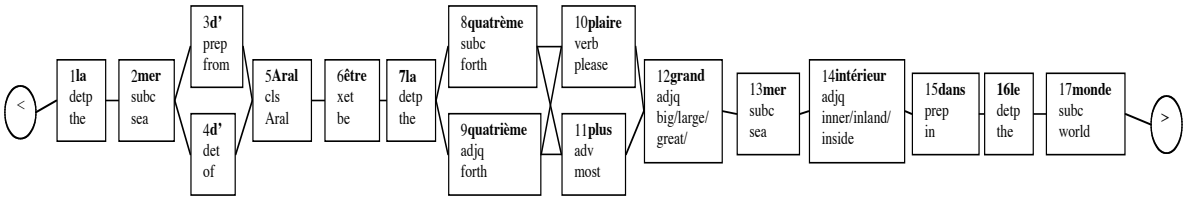


Fig. C-47 Treillis étendu exemple (II)

Et voici une figure pour montrer l'ensemble des liaisons (L_{12}) entre le texte (S_1) et le treillis LMS (S_2).

[18] Ici nous donnons directement le contenu de catégorie et variables grammaticales au lieu des tables pour la facilité de lecture.

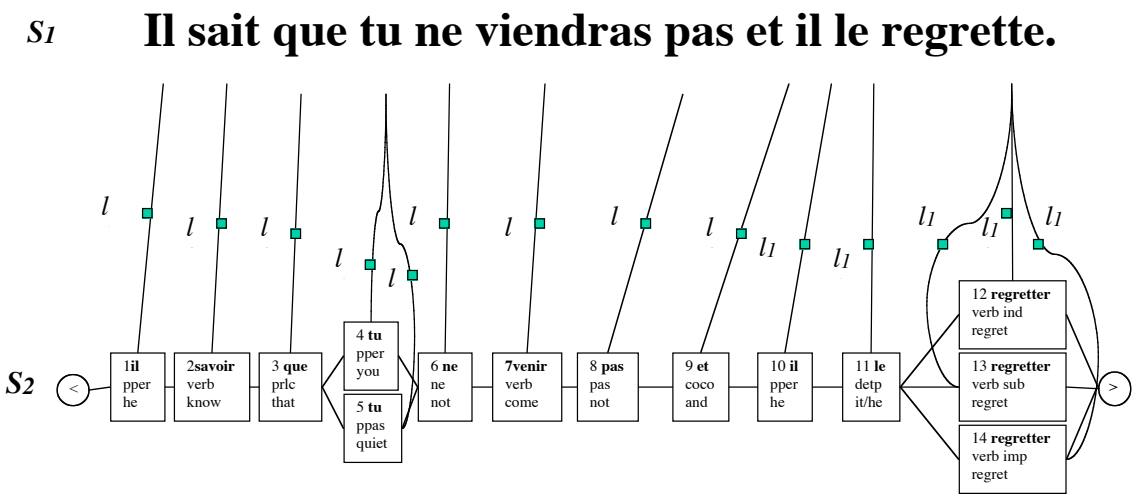


Fig. C-48 L_{12} de l'exemple (I)

Nous avons L_{12} ={l_{12}*}={(identificateur, mot(s) de texte, nœud de treillis)*}={(1, il, 1il), (2, sait, 2savoir), (3, que, 3que), (4, tu, 4tu), (5, tu, 5tu), (6, ne, 6ne), (7,viendras, 7venir), (8, pas, 8pas), (9, et, 9et), (10, il, 10il), (11, le, 10le), (12,regrette, 12regretter), (13, regrette, 13regretter), (14, regrette, 14regretter)}.

3.3 Correspondance entre graphe UNL et arbre UNL

Ici, nous transformons d'abord un graphe UNL en un arbre UNL. Il y a à cela plusieurs avantages:

L'arbre est la représentation la plus exploitée dans le domaine linguistique, il est beaucoup mieux étudié par rapport au graphe.

La correspondance arbre-texte a déjà été beaucoup étudiée, nous pouvons peut-être trouver quelques algorithmes qui peuvent nous aider à créer des liens.

De plus, des transformations graphe UNL _ arbre UNL ont déjà été développées au GETA.

3.3.1 Définition formelle et formalisation possible

Voici quelques définitions formelles entre ces deux couches :

Pour L_{34}

Une liaison l_{34} entre arbre et graphe est de forme :

l_{34} = (identificateur, profil_Nœud, nœud d'arbre+, nœud de graphe) ou

l_{34} = (identificateur, profil_Arc, nœud d'arbre, arc de graphe)

$C_{34} \subseteq L_{34}$ est une correspondance partielle, ce sera vrai pour tout C_{34} par construction.

C_{34} est totale pour le graphe G si et seulement si :

Nœuds-Graphes (C_{34}) = Nœuds (G)

Arcs-Graphes (C_{34}) = Arcs (G)

avec Nœuds-Graphes (C_{34}) = { Nœuds-Graphes (λ)| λ∈ C_{34}}

$$\text{Nœuds-Graphes } (\lambda) = \begin{cases} \varphi & \text{si profil}(\lambda)=\text{Arc} \\ \text{Nœuds-Graphes } (\lambda) & \text{sinon} \end{cases}$$

Tableau C-6 Définitions formelles pour les correspondances graphe-arbre

Nous distinguons deux types de profils de liaisons entre graphe UNL et arbre UNL. Soit un nœud de graphe correspond à un ou plusieurs nœuds d'arbre, ou soit un arc de graphe correspond à un seul nœud d'arbre.

Ensuite nous commençons par l'introduction du passage graphe UNL → arbre UNL.

3.3.2 Description de l'algorithme

L'algorithme de transformation a déjà été développé au GETA, parce que pendant la déconversion UNL_français, tous les graphes UNL sont d'abord transformés en arbres ARIANE-G5 et ensuite ARIANE prend en charge la génération du français.

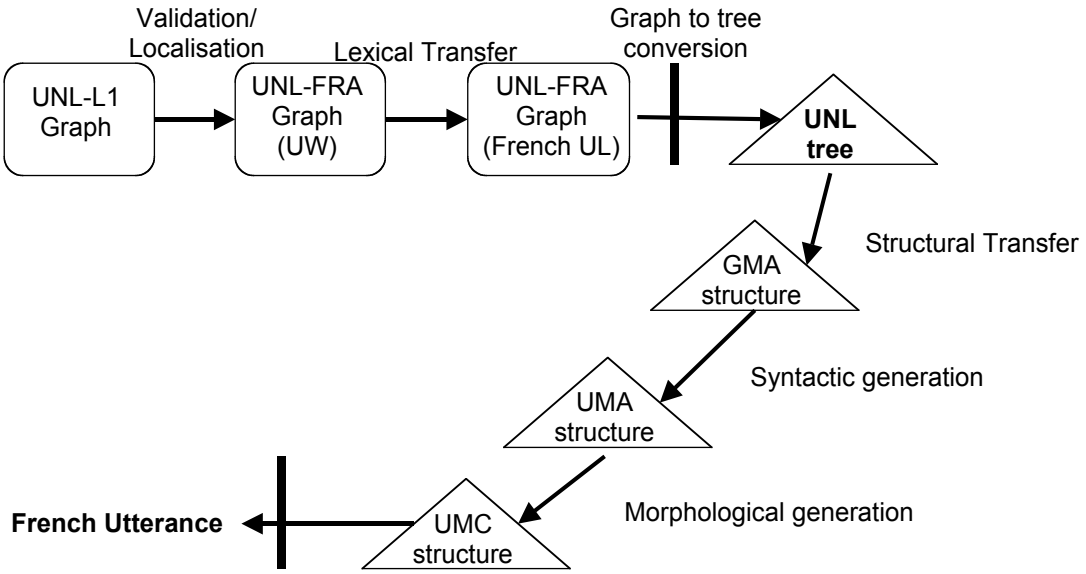


Fig. C-49 Procédure pour la déconversion UNL→français

Un arbre ARIANE-G5 est un arbre général (n-aire ordonné), avec décorations sur les nœuds. Chaque décoration est un ensemble de paires « variable-valeur(s) ». La relation portée par un arc du graphe UNL doit être transformée en étiquette et mise dans un nœud de l'arbre. La transformation graphe_arbre doit conserver l'orientation et les étiquettes du graphe, et aussi les décorations des nœuds.

Cet arbre ARIANE-G5 est donc l'arbre UNL que nous voulons construire. Il contient toutes les informations nécessaires pour reconstruire le graphe UNL.

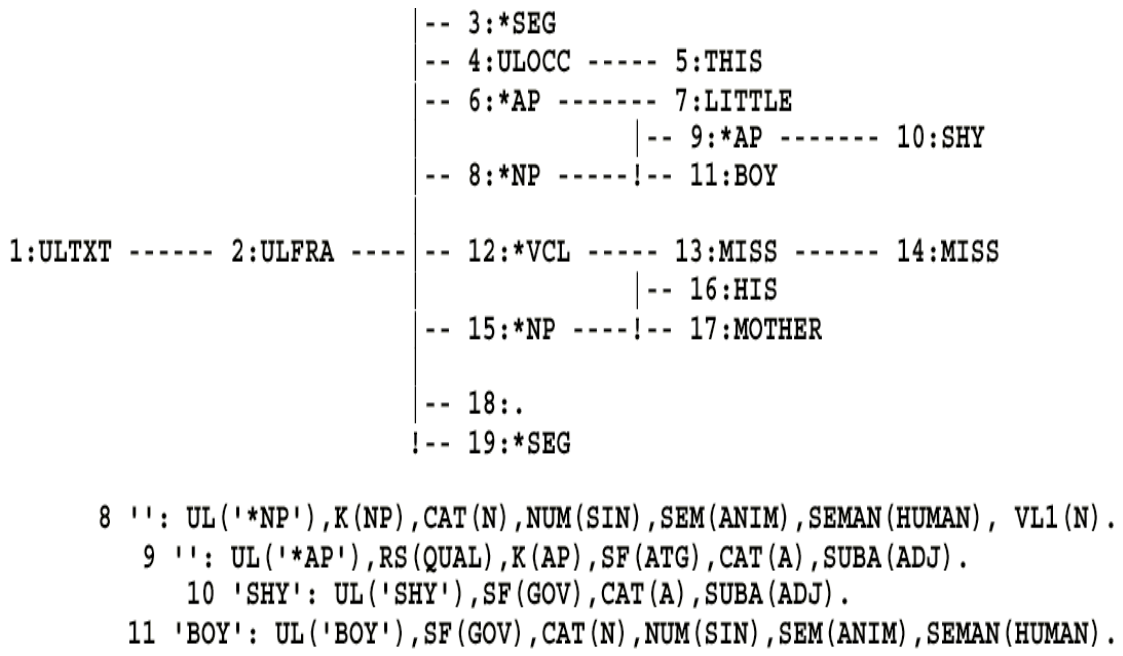


Fig. C-50 Arbre ARIANE-G5 et étiquettes des nœuds

Dans [Sérasset 99] un algorithme pour construire l'arbre ARIANE-G5 par parcours du graphe UNL a été décrit. Cet algorithme prend un graphe UNL, commence par le nœud d'entrée et parcourt le graphe entier. La sortie de cet algorithme est un arbre UNL. Cet algorithme crée plusieurs copies d'un arc qui a plus qu'un arc entrant, et inverse le moins possible d'arcs. Si un arc est inversé, on note dans l'arbre XXREL pour la relation sémantique REL qu'il porte dans le graphe. Le cycle qui est permis dans le graphe UNL mais interdit dans l'arbre est donc cassé.

Plus tard Sérasset a modifié son algorithme pour qu'il puisse gérer un graphe avec scope(s). Maintenant cet algorithme fait partie du déconvertisseur UNL-français.

Voici le détail de cet algorithme :

Soit $\sum$ l'ensemble des nœuds du graphe G, _ l'ensemble des étiquettes (relations du graphe), S_i est un scope dans G, T_i l'arbre généré, et N_i l'ensemble des nœuds de T_i, sn est le numéro de référence de scope. Chaque scope a son pseudo-nœud dans le graphe UNL.

Le graphe $G=\{(a, b, l, sn) \mid a\_\sum, b\_\sum , l\_\_\}=\{\cup S_i\}$ est défini comme un ensemble d'arcs orientés et étiquetés. L'algorithme utilise une liste d'association $A_i=\{(n_G, n_T)| n_G\_\sum, n_T \_N\}$, pour mémoriser la correspondance entre les nœuds de l'arbre et les nœuds du graphe.

Nous avons aussi $G=\cup S_i$, $T=\cup T_i$, $N=\cup N_i$, $A=\cup A_i$, S'={scopes visités}, et A'={arbre visités}.

//construire les scopes et les arbres correspondants
pour (i==0 ; i<NombredeScope ; i++) faire {
soit $e_G \in \sum$ et e est le nœud d'entrée de S_i

```
        e_T ← new nœud d'arbre (e_G , entry)
en T_i ← e_T () ; N_i←e_T ; A_i←(e_G,e_T)
tant que S_i   ∅ faire {
        s'il existe un (a,b,l, sn) en S_i et (a, a_T) ∈ A_i alors
                S_i←S_i\(a,b,l, sn) ;
                b_T ←new nœud d'arbre (b,l) ;
                A_i←A_i∪{(b, b_T)} ;
                soit a_T∈N_i pour que (a,a_T)∈A_i
                en attachant b_T à a_T comme un fils;
        sinon il existe un (a,b,l, sn) en S_i et (b, b_T) ∈ A_i alors
                S_i←S_i\(a,b,l, sn) ;
                b_T ←new nœud d'arbre (a,l^-1) ;
                A_i←A_i∪{(a, a_T)} ;
                soit a_T∈N pour que (b,b_T)∈A_i
                en attachant a_T à b_T comme un fils;
        sinon exit en signalant une erreur (« sous-graphe non-connecté ») ;
        }
        S'←S_i, A'←A_i ; //mémoriser le scope et l'arbre visités
}}
//reconstruire le graphe et l'arbre entier en connectant les morceaux individuels
connecter tous les pseudo-nœuds dans A_0  avec leurs arbres ;
sortir A_0 ;
}
```

Fig. C-51 algorithme de transformation d'un graphe UNL en un arbre UNL (d'après G. Sérasset)

Voici une figure illustrant la duplication d'un nœud et l'inversion d'un arc ($z \rightarrow z^{-1}$), la relation étant mise dans l'étiquette du nœud de l'arbre correspondant au nœud d'arrivée de l'arc (ou au nœud de départ en cas d'inversion). Le nœud « a » est le nœud d'entrée.

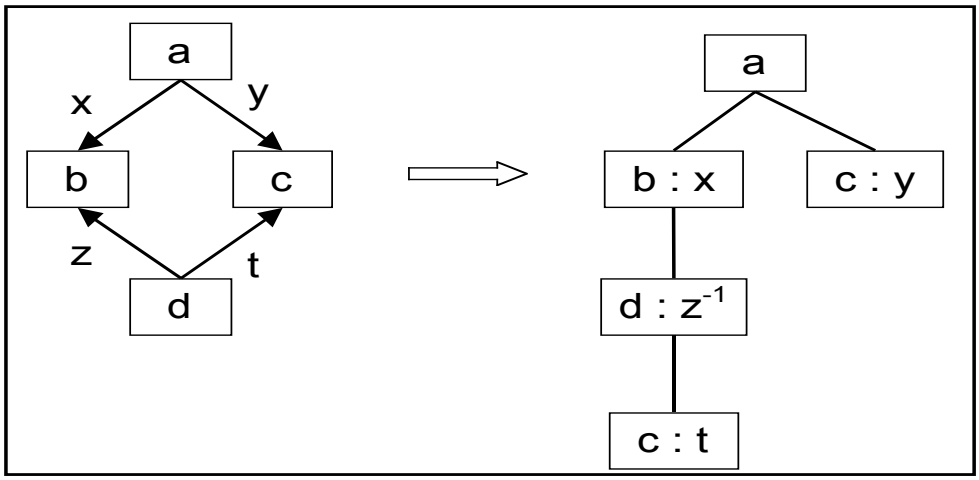


Fig. C-52 Inversion d'un arc ($z \rightarrow z^{-1}$) et duplication d'un nœud (c)

Voici quelques exemples de cette transformation.

3.3.2.1 Graphe simple

Il y a deux cadres dans la figure ci-dessous : celui de gauche contient le graphe UNL. Celui de droite comprend l'arbre ARIANE-G5 correspondant avec un transfert lexical vers le français. Plus bas, on voit la représentation textuelle de l'arbre et les décorations des nœuds d'arbre. Dans l'algorithme de Sérasset, pour faciliter le calcul on stocke les restrictions sur des fils d'un nœud. La racine de l'arbre ARIANE est toujours « ULTXT », puis « ULFRA » (le transfert lexical vers le français).

[S:1]
{unl}
agt(catch(icl>do).@entry.@present,cat(icl>feline).@def)
obj(catch(icl>do).@entry.@present,mouse(icl>rat))
{/unl}
{fr}Le chat attrape une petite souris.{/fr}
[/S]

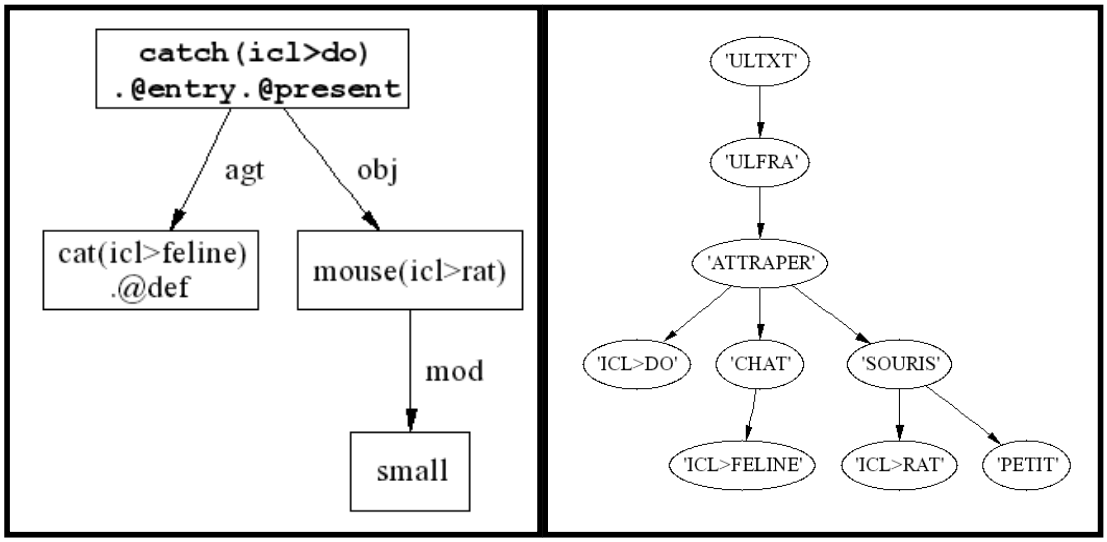


Fig. C-53 Transformation d'un graphe UNL simple en un arbre ARIANE

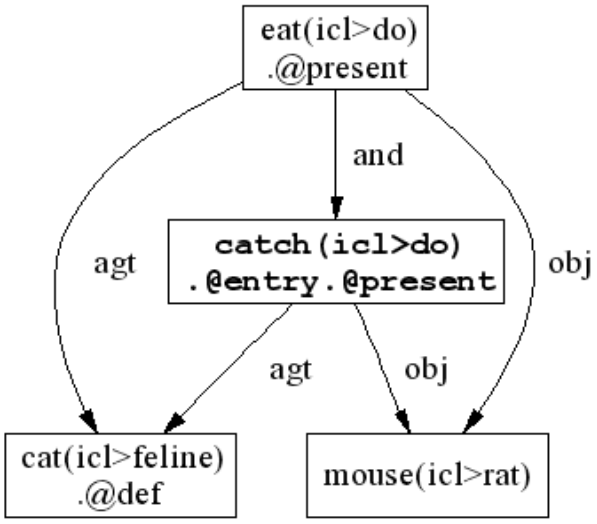
1:'ULTXT'(2:'ULFRA'(3:'ATTRAPER'(4:'ICL>DO',5:'CHAT'(6:'ICL>FELINE'),7:'SOURIS'(8:'ICL>RAT'))))

1 'ULTXT' : UL('ULTXT').
2 'ULFRA' : UL('ULFRA').
3 'ATTRAPER' :
UL('ATTRAPER'),AUX(AVOIR),CAT(CATV),INST(0),VAL1(GN),VARUNL(ENTRY,PRESENT).
4 'ICL>DO' : UL('ICL>DO'),RESTR(1).
5 'CHAT' :
UL('CHAT'),CAT(CATN),GNR(MAS),INST(1),RSUNL(AGT),VARUNL(DEF).
6 'ICL>FELINE' : UL('ICL>FELINE'),RESTR(1).
7 'SOURIS' : UL('SOURIS'),CAT(CATN),GNR(FEM),INST(2),RSUNL(OBJ).
8 'ICL>RAT' : UL('ICL>RAT'),RESTR(1).

3.3.2.2 Graphe non arborescent

Quand le graphe n'est pas arborscent, chaque nœud pointé par plus d'un arc est dupliqué dans l'arbre, et les nœuds d'arbre correspondant à un même nœud de graphe ont la même marque d'instance dans leurs décorations, comme nous le voyons ici « inst(1) » pour « cat(icl>feline) » et « inst(2) » pour « mouse(icl>rat) ». INST code donc directement dans un nœud (non auxiliaire comme « ULFRA ») de l'arbre de l'arbre le nœud du graphe dont il provient. Il y a une relation inverse après la transformation, donc le nœud de l'arbre « manger » porte la relation « XXAND » dans sa décoration.

[S :1]
{unl}
agt(catch(icl>do).@entry.@present,cat(icl>feline).@def)
obj(catch(icl>do).@entry.@present,mouse(icl>rat))
and(eat(icl>do).@present,catch(icl>do).@entry.@present)
agt(eat(icl>do).@present,cat(icl>feline).@def)
obj(eat(icl>do).@present,mouse(icl>rat))
{/unl}
{fr} Le chat attrape une souris et la mange.{/fr}
[/S]



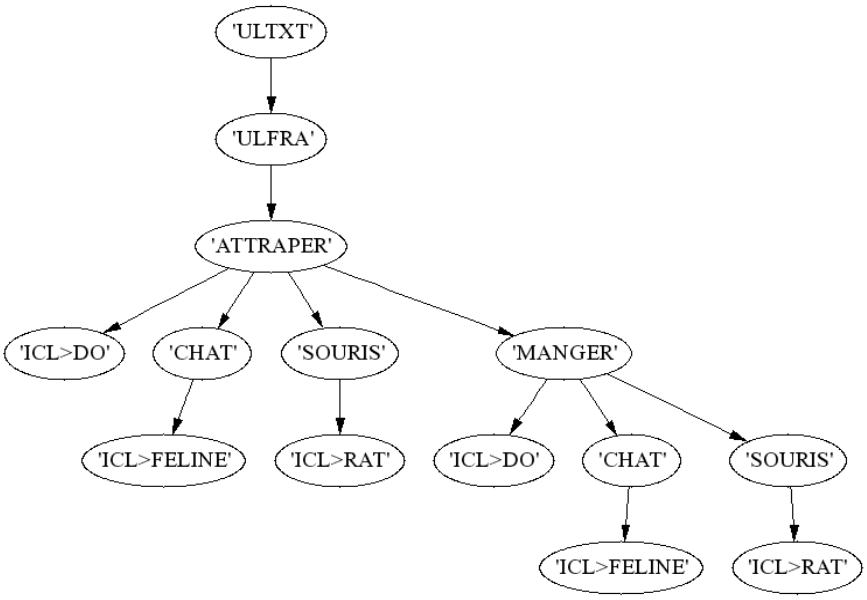


Fig. C-54 Transformation d'un graphe UNL non arborescent en un arbre ARIANE

1:'ULTXT'(2:'ULFRA'(3:'ATTRAPER'(4:'ICL>DO',5:'CHAT'(6:'ICL>FELINE'),7:'S
OURIS'(8:'ICL>RAT'),9:'MANGER'(10:'ICL>DO',
11:'CHAT'(12:'ICL>FELINE'),13:'SOURIS'(14:'ICL>RAT')))))

1 'ULTXT' : UL('ULTXT').
2 'ULFRA' : UL('ULFRA').
3 'ATTRAPER' :
UL('ATTRAPER'),AUX(AVOIR),CAT(CATV),INST(0),VAL1(GN),VARUNL(EN
TRY,PRESENT).
4 'ICL>DO' : UL('ICL>DO'),RESTR(1).
5 'CHAT' :
UL('CHAT'),CAT(CATN),GNR(MAS),INST(1),RSUNL(AGT),VARUNL(DEF).
6 'ICL>FELINE' : UL('ICL>FELINE'),RESTR(1).
7 'SOURIS' : UL('SOURIS'),CAT(CATN),GNR(FEM),INST(2),RSUNL(OBJ).
8 'ICL>RAT' : UL('ICL>RAT'),RESTR(1).
9 'MANGER' :
UL('MANGER'),AUX(AVOIR),CAT(CATV),INST(3),RSUNL(XXAND),VAL1(GN
),VARUNL(PRESENT).
10 'ICL>DO' : UL('ICL>DO'),RESTR(1).
11 'CHAT' :
UL('CHAT'),CAT(CATN),GNR(MAS),INST(1),RSUNL(AGT),VARUNL(DEF).
12 'ICL>FELINE' : UL('ICL>FELINE'),RESTR(1).
13 'SOURIS' : UL('SOURIS'),CAT(CATN),GNR(FEM),INST(2),RSUNL(OBJ).
14 'ICL>RAT' : UL('ICL>RAT'),RESTR(1).

3.3.2.3 Graphe avec scope

Chaque scope sauf le scope initial (englobant) est représenté par un pseudo-nœud « S+numéro de scope ». Chaque nœud de graphe dupliqué reçoit la même valeur pour la variable « inst »[19]. Le principe de la duplication de nœud est :

> si n+2 nœuds distincts du graphe ont la même UW, on crée n+2 ensemble de nœuds de l'arbre, avec la même marque d'instance,
>
> à l'intérieur d'un ensemble, tous les nœuds de l'arbre sont des feuilles, sauf un, le premier créé, dont ils sont les clones, et qui deviennent le sous-arbre image du sous-graphe accédé par ce nœud.

Au moment du transfert lexical, si on trouve plus d'un lemme correspondant dans le dictionnaire, un pseudo-nœud « AMBIG » est inséré dans l'arbre, puis tous les lemmes candidats y sont attachés.

[S:1]
{unl}
obj(flow(icl>occur).@entry.@not.@past,river.@def.@pl)
man(flow(icl>occur).@entry.@not.@past,almost)
rsn(flow(icl>occur).@entry.@not.@past,:01)
obj:01(block(icl>do).@entry.@past,river.@def.@pl)
agt:01(block(icl>do).@entry.@past,dam.@pl)
{/unl}
{fr} Bloquées par des barrages, les rivières ne coulaient presque plus. {/fr}
[/S]

[19] La distribution de la marque d'instance est faite seulement pour les UW différentes (et pour le pseudo-nœud AMBIG, présent si l'on a plusieurs lemmes candidats), et pour chaque pseudo-nœud scope. La marque d'instance ne sera pas distribuée aux nœuds d'arbre qui portent les restrictions d'une UW ou les lemmes candidats.

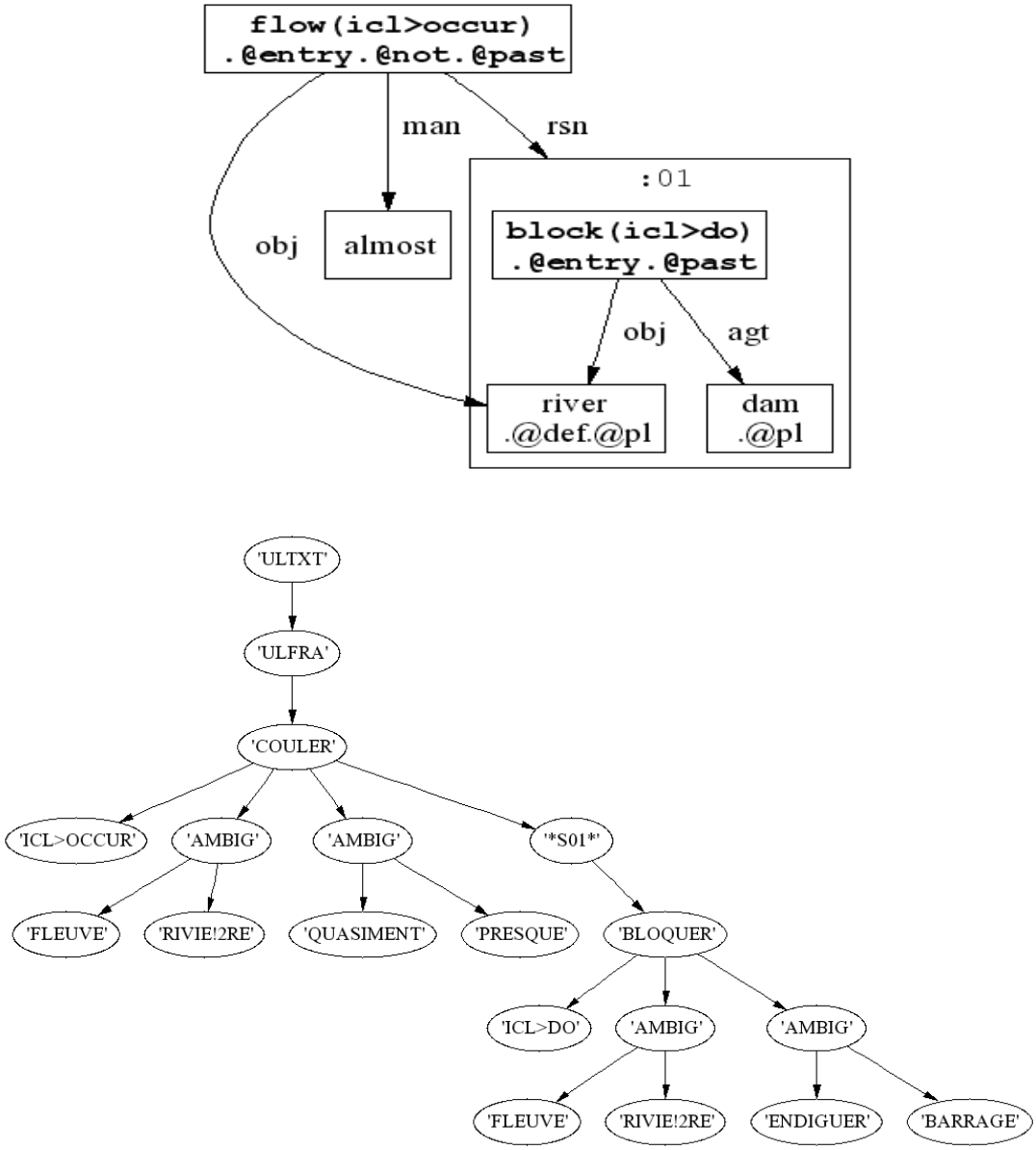


Fig. C-55 Transformation d'un graphe UNL avec scope (en haut) en un arbre ARIANE (en bas)

1:'ULTXT'(2:'ULFRA'(3:'COULER'(4:'ICL>OCCUR',5:'AMBIG'(6:'FLEUVE',7:'RIVIE!2RE'),8:'AMBIG'(9:'QUASIMENT',10:'PRESQUE'),
11:'*S01*'(12:'BLOQUER'(13:'ICL>DO',14:'AMBIG'(15:'FLEUVE',16:'RIVIE!2RE'),17:'AMBIG'(18:'ENDIGUER',19:'BARRAGE'))))))

1 'ULTXT' : UL('ULTXT').
2 'ULFRA' : UL('ULFRA').
3 'COULER' :
UL('COULER'),AUX(AVOIR),CAT(CATV),INST(3),VARUNL(ENTRY,NOT,PAST).
4 'ICL>OCCUR' : UL('ICL>OCCUR'),RESTR(1).

5 'AMBIG' : UL('AMBIG'),AMBIG(0),INST(1),RSUNL(OBJ),VARUNL(DEF,PL).
6 'FLEUVE' : UL('FLEUVE'),AMBIG(1),CAT(CATN),GNR(MAS),N(NC).
7 'RIVIE!2RE' : UL('RIVIE!2RE'),AMBIG(1),CAT(CATN),GNR(FEM),N(NC).
8 'AMBIG' : UL('AMBIG'),AMBIG(0),INST(4),RSUNL(MAN).
9 'QUASIMENT' : UL('QUASIMENT'),AMBIG(1),CAT(CATADV).
10 'PRESQUE' : UL('PRESQUE'),AMBIG(1),CAT(CATADV).
11 '*S01*' : UL('*S01*'),INST(5),RSUNL(RSN).
12 'BLOQUER' :
UL('BLOQUER'),AUX(AVOIR),CAT(CATV),INST(0),RSUNL(GRP),VAL1(GN),
VARUNL(ENTRY,PAST).
13 'ICL>DO' : UL('ICL>DO'),RESTR(1).
14 'AMBIG' : UL('AMBIG'),AMBIG(0),INST(1),RSUNL(OBJ),VARUNL(DEF,PL).
15 'FLEUVE' : UL('FLEUVE'),AMBIG(1),CAT(CATN),GNR(MAS),N(NC).
16 'RIVIE!2RE' : UL('RIVIE!2RE'),AMBIG(1),CAT(CATN),GNR(FEM),N(NC).
17 'AMBIG' : UL('AMBIG'),AMBIG(0),INST(2),RSUNL(AGT),VARUNL(PL).
18 'ENDIGUER' :
UL('ENDIGUER'),AMBIG(1),AUX(AVOIR),CAT(CATV),VAL1(GN).
19 'BARRAGE' : UL('BARRAGE'),AMBIG(1),CAT(CATN),GNR(MAS),N(NC).
[/D]

Si l'utilisateur réalise cette transformation sur le serveur de déconversion UNL-français, un avertissement est affiché quand l'ambiguïté lexicale apparaît :

!!!++++++++++++++++++++ Graphe -> Arbre ++++++++++++++++++++++++++
WARN [DICTIONARY] : l'UW "river" a plusieurs traductions dans le dictionnaire :
[DICTIONARY] : --> fleuve : CAT(CATN),GNR(MAS),N(NC)
[DICTIONARY] : --> rivière : CAT(CATN),GNR(FEM),N(NC)
WARN [DICTIONARY] : l'UW "dam" a plusieurs traductions dans le dictionnaire :
[DICTIONARY] : --> endiguer : AUX(AVOIR),CAT(CATV),VAL1(GN)
[DICTIONARY] : --> barrage : CAT(CATN),GNR(MAS),N(NC)
WARN [DICTIONARY] : l'UW "almost" a plusieurs traductions dans le dictionnaire
:
[DICTIONARY] : --> quasiment : CAT(CATADV)
[DICTIONARY] : --> presque : CAT(CATADV)
!!!++

Nous n'utilisons pas directement le résultat de cette transformation. Nous voulons garder toutes les informations liées d'une UW dans un seul nœud d'arbre. Puis nous voulons garder tous les lemmes candidats d'une UW dans son nœud d'arbre. Nous devons changer un peu la structure de l'arbre sorti.

3.3.3 Structure de données et calcul possible

Nous présentons le graphe UNL (S_4) avant l'arbre UNL (S_3), parce que l'arbre UNL est produit à partir du graphe UNL.

Un graphe UNL G est un ensemble d'arcs. Chaque arc Arc se compose de deux nœuds (neoudgraphe) et d'une relation. N_g est l'ensemble de nœuds dans ce graphe. Nous avons donc :

$G = \{ Arc_i \}_{i\in[1\ldots \text{nombre d'arcs}]} = \{(id\_rel, relation, noeudgraphe_i , noeudgraphe_j , id\_niveau)^*\}_{i,j\in[1\ldots \text{nombre d'arcs}],\ i\neq j}$ ·

Chaque relation appartient à {agt, and, aoj, bas, ……via}

Un nœudgraphe se compose d'un identificateur, d'une référence du sous-graphe auquel il appartient, d'un Headword, d'une liste de restrictions et d'une liste d'attributs.

$noeudgraphe_i$ = (id_noeudgr, subgraphref, UW_i , attribut*) = (id_noeudgr, subgraphref, HW_i , restriction* , attribut*)

chaque attribut appartient à {.@past, .@present, .@future, .@begin, …. , .@surprised}

$N_g = \{noeudgraphe_i\}_{i\in[1\ldots \text{nombre de nœuds}]}$ ·

Dans notre exemple (I), on a:
{unl}
1 : agt(regret(icl>do).@entry, he(icl>human))
2 : and(regret(icl>do).@entry, know(agt>human, obj>event))
3 : agt(know(agt>human, obj>event), he(icl>human))
4 : obj(regret(icl>do).@entry, :01)
5 : obj(know(agt>human, obj>event), :01)
6 : agt:01(come.@entry.@future.@not, you)
{/unl}
{fr}il sait que tu ne viendras pas et il le regrette.{/fr}

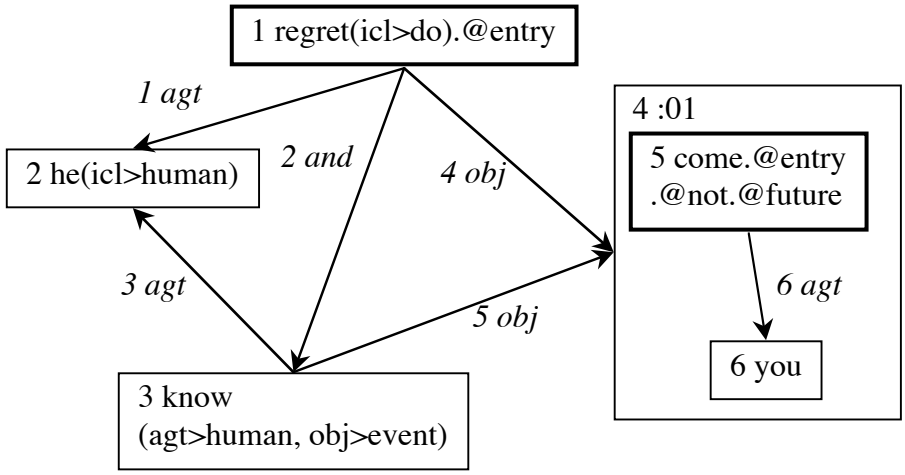


Fig. C-56 Graphe UNL avec les arcs et les nœuds numérotés exemple (I)

Dans notre exemple (II), on a:
{unl}
1 : nam(sea:01.@def, Aral)
2 : aoj(sea:02.@def.@entry.@past, sea:01.@def)
3 : mod(sea:02.@def.@entry.@past, inland(mod<thing))
4 : mod(sea:02.@def.@entry.@past, fourth(mod<thing))
5 : mod(sea:02.@def.@entry.@past, large)

6 : man(large, most)
7 : scn(large, world.@def)
{/unl}

N_g = {(1, 0, sea:01, -, .@def), (2, 0, Aral, -, -), (3, 0, sea :02, -, .@def.@entry.@past), (4, 0, inland, (mod<thing), -), (5, 0, fourth, (mod<thing), -), (6, 0, large, -, -), (7, 0, most, -, -), (8, 0, world, -, .@def) }
G = $\{Arc_i\}_{i\in[1\ldots \text{nombre d'arcs}]}$ = {(1, nam, 1, 2, 00), (2, aoj, 3, 1, 00), (3, mod, 3, 4 , 00), (4, mod , 3, 5, 00), (5, mod, 3, 6, 00), (6, man, 6, 7, 00), (7, scn, 6, 8, 00)}

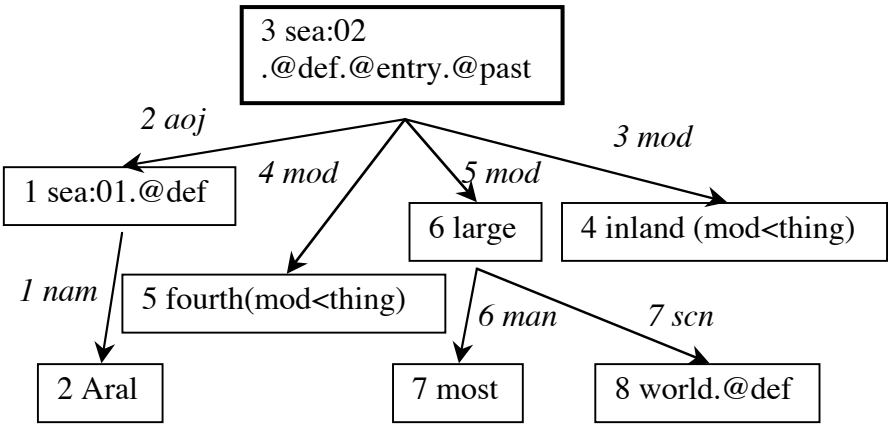


Fig. C-57 Graphe UNL avec les arcs et les nœuds numérotés exemple (II)

Un arbre UNL A est un ensemble de $noeudarbre_i$. Chaque $noeudarbre_i$ a pour composants un lemme ($lemar_i$), une décoration, son père, et la liste de ses fils. La décoration $decoarbre_i$ est un couple (restriction, attribut*). Chaque nœud dans l'arbre n'a qu'un seul père mais il peut avoir un ou plusieurs fils.

$noeudarbre_i$ = (id_noeudar, $lemar_i$, $decorationarbre_i$, $npere_i$, $nfils_i$)

Dans notre exemple (II), on a:
A = N_a = $\{noeudarbre_i\}_{i\in[1\ldots \text{nombre de nœuds}]}$
= {(id_noeudar, lemar, restriction, attribut*, relation, npere, nfils)*}
= {(1, sea, -,.@def.@entry.@past,-, -, 2/3/4/5), (2, sea, , -, .@def, aoj, 1, 6), (3, fourth, (mod<thing), -, mod, 1,-), (4, large, -, -, mod, 1, 7/8), (5, inland, (mod<thing), -,mod, 1, -), (6, Aral, -, -, nam,2, -), (7, most, -,man, -, 4, -), (8, world, -, .@def, scn,4, -)}

Pour chaque nœud $noeudarbre_i$ nous consultons le dictionnaire UNL-français, et ajoutons à ce nœud les lemmes français trouvés et compatibles. Nous obtenons un $noeudarbre_i$ francisé. Nous appelons ce noeudarbre francisé un noeudarbreetendu.

Voici un exemple du format de dictionnaire UNL-français (voir Annexe E pour plus de détails) :

[abominablement]{CAT(CATADV)}"abominably(icl>manner)";
[abomination]{CAT(CATN),GNR(FEM),N(NC)}"abomination";
[abominer]{AUX(AVOIR),CAT(CATV),VAL1(GN)}"abominate";

[abortif]{CAT(CATADJ)}"abortive(icl>state)";

On utilise une table de compatibilité analogue à celle utilisée pour construire le treillis étendu. Voici un extrait de cette table. (Ici, on utilise les catégories du dictionnaire UNL-FR, qui sont légèrement différentes de celles de l'AMS PILAF.)

Dictionnaire français		**UNL**
catégories		*restriction*
CATADV	Adverbe	(icl>how)
CATN	substantif commun	(icl>thing)
CATADJ	Adjectif qualificatif	(mod<thing)/(aoj>thing)
CATV	Verbe	(icl>do)/(icl>occur)/(icl>state)

Tableau C-7 Table de compatibilité pour arbre étendu

Comme nous l'avons fait avec les nœuds de treillis, nous allons numéroter tous les attributs et restrictions d'UNL et nous aurons deux tables de variables booléennes pour tous les nœuds d'arbre. Il y a 41 relations, 72 attributs dans les spécifications d'UNL.

Quant aux restrictions, elles peuvent être très compliquées : chaque auteur de graphes UNL écrit les restrictions dans son propre style, sans ou avec la référence à la KB. Il y a tout de même des règles pour ça (cf. FB2004). Il nous faut simplifier les restrictions. Nous utilisons les quatre catégories principales de la KB : chaque fois qu'on rencontre une restriction compliquée, on remonte à la racine de la KB en suivant la hierarchie. On a donc 4 restrictions principales qui correspondent aux catégories nom, verbe, adjectif et adverbe.

Voici un exemple. Voici les trois tables du nœud d'arbre « 3know(agt>human, obj>event) and » de Fig. C-58 :

(le nœud « know » appartient à la catégorie du verbe)

numéro	0	1	2	3	
tab_restriction	icl>do	icl>thing	mod<thing	icl>how	
valeur Boolean	1	0	0	0	*longueur=4*

(le nœud « know » n'a pas porté d'attribut, donc toutes les valeurs sont zéro)

numéro	0	1		71	
tab_attribut	@ability	@admire	……..	@yet	
valeur Boolean	0	0	…….	0	*longueur=72*

(la relation que le nœud « know » porte est « and »)

numéro	0	1	2		40	
tab_relation	agt	and	aoj	……	via	
valeur Booelan	0	1	0	…….	0	*longueur=41*

On a alors la structure :
noeudarbreetendu = (id_noeudar, lemar, tab_restriction, tab_attribut, relation, (lemmefrançais, catfrançais)*, npere, nfils)

Nous définissons enfin l'arbre francisé A_f et l'ensemble de noeudarbreetendu N_{af} .

$A_f = N_{af} = \{noeudarbreetendu_i\}$ $i \in [1\ldots$ nombre de nœuds$]$

Dans nos exemples, les arbres francisés sont :

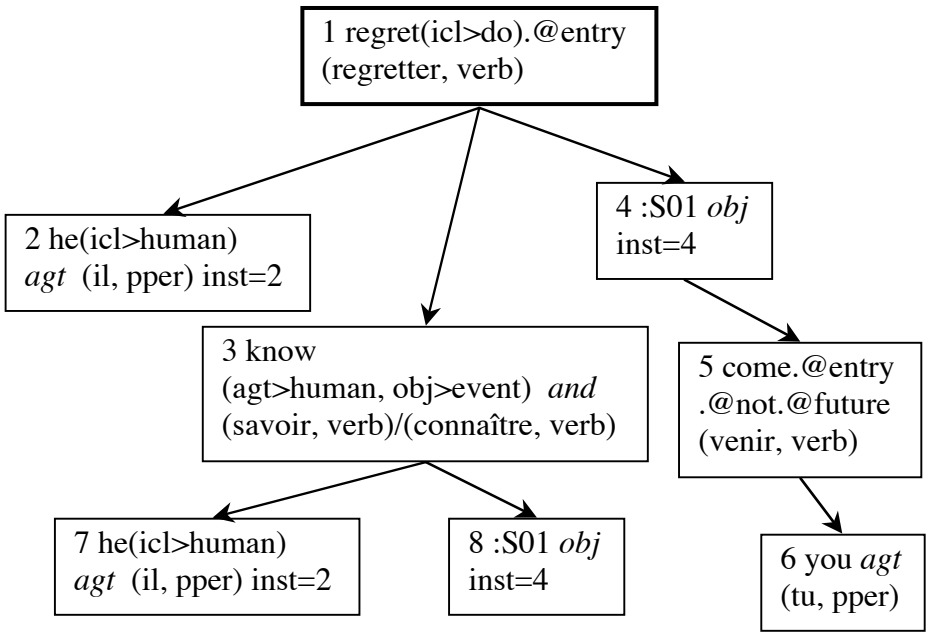


Fig. C-58 Arbre UNL francisé numéroté exempls (I)

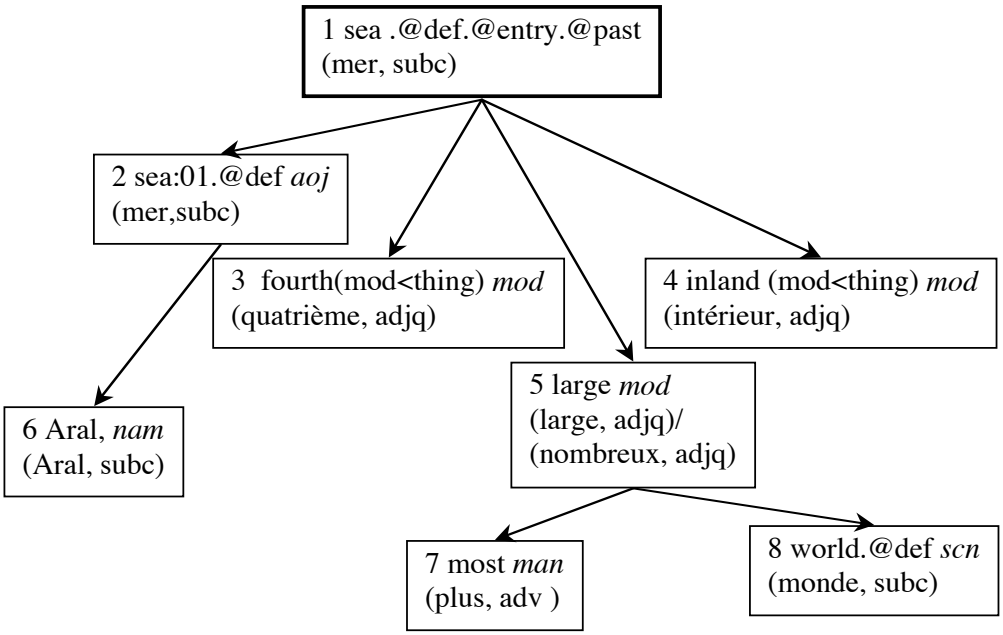


Fig. C-59 Arbre UNL francisé numéroté exempls (II)

Dans l'exemple (II), nous avons:
$A_f = N_{af} = \{noeudarbreetendu_i\}$ $i \in [1\ldots$ nombre de nœuds$]$ = {(id_noeudar, lemar, tab_restriction, tab_attribut, tab_relation, (lemmefrançais, catfrançais)*, npere, nfils)*}

= {(1, sea, -,.@def.@entry.@past, (mer,subc)/(marin, adjq), -, 2/3/4/5), (2, sea, , -, .@def, aoj, (mer,subc)/(marin, adjq), 1, 6), (3, fourth, (mod<thing), -, , mod, (quatrième, subc)/(quatrième, adjq), 1,-), (4, large, -, -, mod, (large, adjq)/(nombreux, adjq), 1, 7/8), (5, inland, (mod<thing), -, mod, (intérieur, adjq), 1, -), (6, Aral, -, -, nam, (Aral, subc), 2, -), (7, most, -, -,man, (plus, adv), 4, -), (8, world, -, .@def, scn, (monde, subc), 4, -)}

La construction de L_{34} est faite au moment où on construit l'arbre UNL.

Les deux figures suivantes montrent l'ensemble des liaisons construites entre un arbre UNL et un graphe UNL. Nous utilisons un triangle pour représenter une liaison de type nœud-are et un carré pour une liaison de type nœud-nœud.

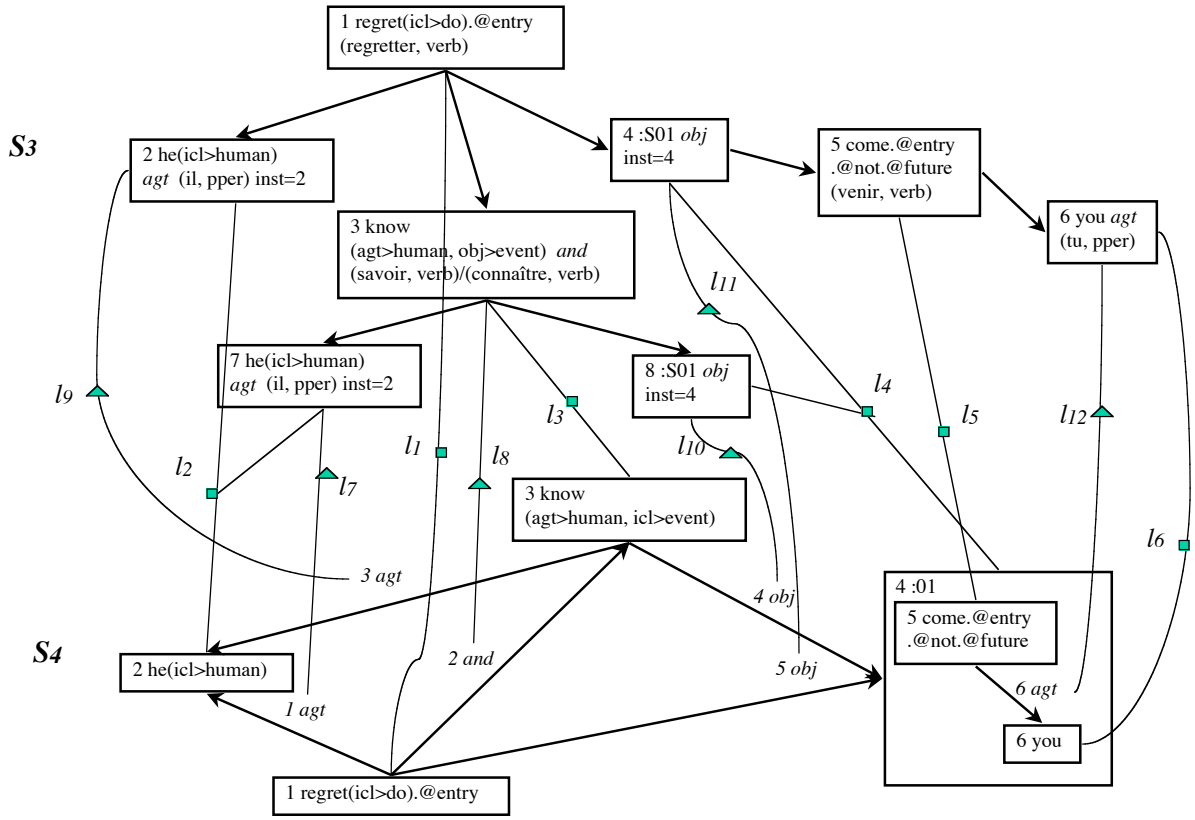


Fig. C-60 L_{34} de l'exemple (I)

Nous avons donc L_{34} ={l_{34}*}={(identificateur, profil de liaison, nœud(s) d'arbre, nœud ou arc de graphe)={(1,nœud,1,1), (2, nœud, 2/7,2), (3, nœud,3,3), (4,nœud,4/8,4), (5,nœud,5,5), (6,nœud,6,6), (7,arc,7,1agt), (8,arc,3,2and), (9,arc,2,3agt), (10,arc,8,4obj), (11,arc,4,5obj), (12,arc,6,6agt)}

Dans le prochain exemple, les liaisons sont plus simples, il n'y a que des liaisons 1_1.

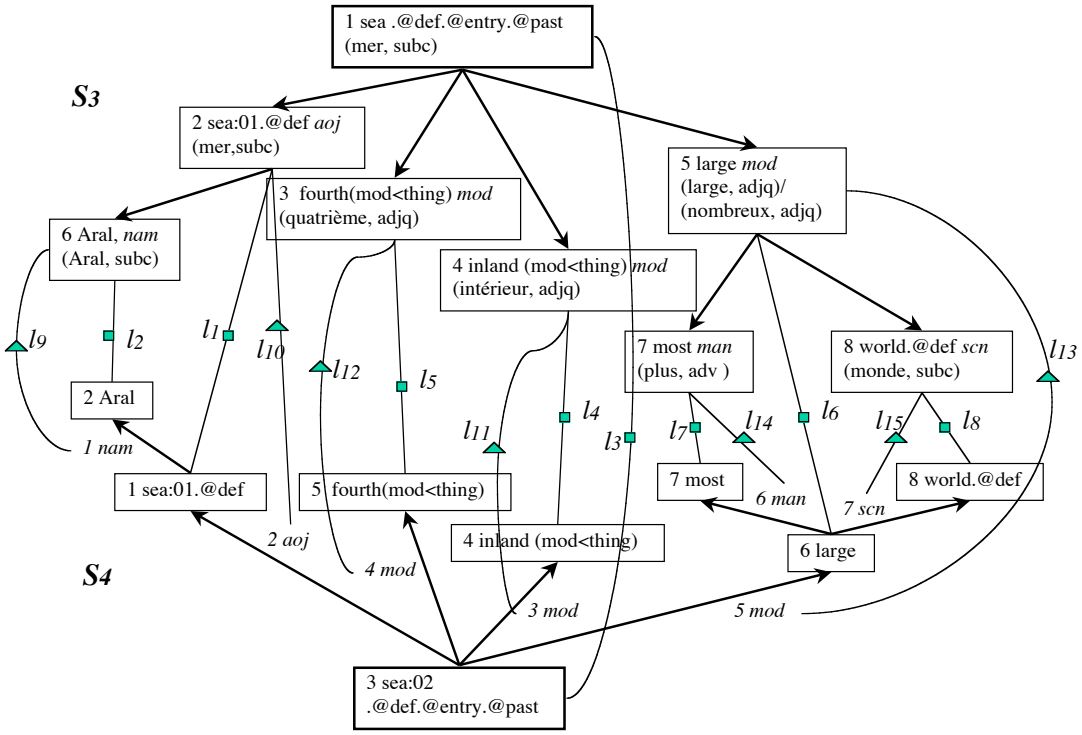


Fig. C-61 L_{34} de l'exemple (II)

Nous avons pour l'exemple (II) :

L_{34} ={l_{34}*}={(identificateur, profil de liaison, nœud(s) d'arbre, nœud ou arc de graphe)={(1,nœud,2,1), (2,nœud,6,2), (3,nœud,1,3), (4,nœud,4,4), (5,nœud,3,5), (6,nœud,5,6), (7,nœud,7,7), (8,nœud,8,8), (9,arc,7,1man), (10,arc,2,2aoj), (11,arc,4,3mod), (12,arc,3,4mod), (13,arc,5,5mod), (14,arc,7,6man), (15,arc,8,7scn)}

3.4 Correspondance entre arbre UNL et treillis LMS

Ici, nous pouvons associer à la chaîne un treillis, grâce à la structure morpho-syntaxique du texte de surface. En effet, comme nous l'avons montré dans la section A.3.3.5, la sortie d'un AMS, d'un segmenteur, ou d'un baliseur (tagger) est normalement un treillis. Après la construction des liaisons texte - treillis LMS (L_{12}) et graphe UNL - arbre UNL (L_{34}), nous cherchons maintenant la construction des liaisons treillis LMS – arbre UNL (L_{23}).

Il y a trois tâches dans cette étape.

(1) Créer une trajectoire provisoire sur le treillis selon les liaisons lexicales construites et vérifier s'il existe un nœud de treillis ou d'arbre qui est pointé (utilisé) plus d'une fois par ces liaisons lexicales. Si non, on a trouvé la meilleure trajectoire, et on va à l'étape (3).

(2) Si oui, ajuster la forme de l'arbre (en modifiant seulement la précédence linéaire) pour qu'il y ait le moins possible de croisements de liaisons entre l'arbre et le chemin choisi, c'est-à-dire trouver un meilleur ordre « horizontal » sur l'arbre,

(3) Enrichir la correspondance L_{23}, c'est-à-dire créer autant de liaisons que possible entre l'arbre UNL et le chemin choisi dans le treillis.

3.4.1 Définition formelle et formalisation possible

Voici les définitions formelles de notre problème :

Pour L_{23} , on cherche une C « correcte » (Nœuds-Treillis (C) ⊆ Trajectoire) qui maximise une certaine fonction f, avec

f(C) diminue si le nombre de croisements augmente,

f(C) augmente si le nombre de liaisons augmente,

f(C) augmente si le poids des liaisons augment.

On pose :

Poids(λ) = g(type(λ))*nb_liaison(λ).

*(il faut consulter la table de compatibilité pour décider g(type(λ)), on suppose 0<g≤5)

On pose pour f : f(C, Ordre sur arbre) = f(C, O) = α*Poids(C) - β*Croisement(C,O).

*(On cherche un meilleur couple (C,O), O abrège arbre muni de l'ordre O et on suppose β=10)

$$\text{Poids}(C) = \sum_{\lambda \in C} \text{Poids}(\lambda) .$$

Tableau C-8 Définition formelle de la correspondance arbre-treillis

3.4.2 Étude préliminaire du problème

Ajuster l'ordre d'un arbre pour qu'il corresponde au mieux à un chemin dans le treillis est un problème combinatoire.

Supposons que nous avons un arbre très simple de trois nœuds (1, 2, et 3) et deux niveaux, avec la racine (1) et ses deux fils. Le nombre de liaisons différentes est 3!:: (1(2,3)), (1(3,2)), ((2),1,(3)), ((3), 1,(2)), ((2,3)1), et ((3,2)1).

Selon ce calcul, nous pouvons déduire le nombre de liaisons différentes d'un arbre :

Notons $Nb(N_i)$ est le nombre d'ordres différents d'un arbre dont la racine est le nœud N_i, et $\_(N_i)$ le nombre de fils du nœud N_i .

Supposons que nous avons un arbre dont la racine est N_0 , avec $\_(N_0)=k$.

Nous avons :

$$Nb(N_0) = (\_(N_0)+1)!(Nb(N_1)+Nb(N_2)+\ldots+Nb(N_k) = (k+1)!\sum_{i=1}^{k} Nb(N_i)$$

Sans contraintes, le nombre de correspondances arbre-treillis peut être gigantesque. Mais les contraintes de correspondances lexicales réduisent en fait l'espace de recherche de façon considérable, et l'exploration de l'arbre de recherche correspondant est possible en temps raisonnable.

L'exemple suivant (tiré du corpus « La main à la pâte ») montre une phrase, déconvertie d'un graphe UNL assez compliqué. On constate que le texte déconverti du graphe UNL correspond généralement assez bien à son graphe UNL, bien que la qualité de déconversion ne soit pas parfaite.

> Le graphe UNL vient du texte français « ce site est le vôtre, nous comptons donc sur vous pour le faire vivre et pour l'enrichir. »
>
> {unl}
> mod:01(website:1.@entry,your)
> aoj:01(website:1.@entry,website:2)
> mod:01(website:2,this(mod<thing))
> mod(:01.@entry,:02)
> obj:02(therefore.@entry,rely upon(icl>do))
> agt:02(rely upon(icl>do),we)
> obj:02(rely upon(icl>do),you)
> pur:02(rely upon(icl>do),:03)
> and:03(enrichment.@entry,animation)
> mod:02(:03,website:3)
> {/unl}

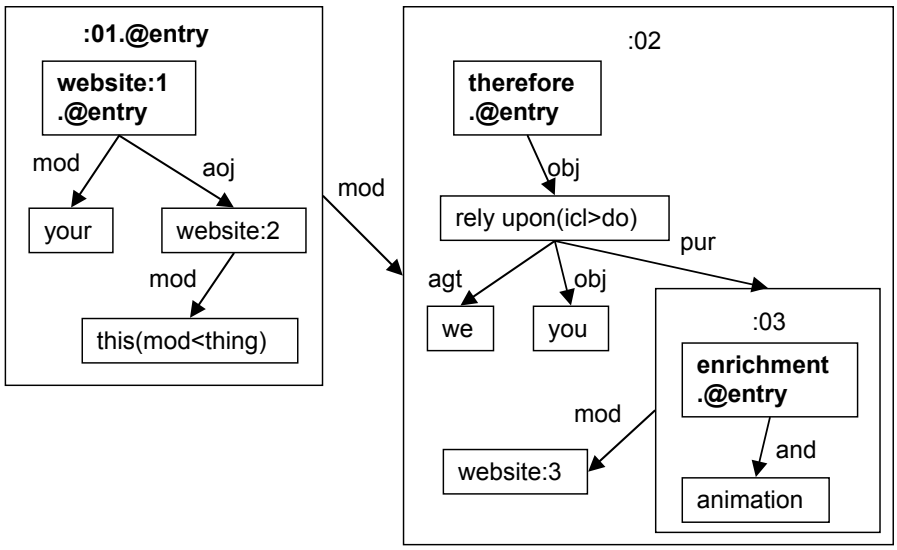


Fig. C-62 Un graphe UNL assez compliqué

Ce graphe est loin d'être parfait, ce dont on s'aperçoit en lisant les déconversions obtenues.

français déconverti : « Ce site web est votre site web aussi qu'on vous compte pour l'animation et un enrichissement de site web. »

russe déconverti : « ___ _____-_____ - ____ _____-_____ _____ __ _____ __ ___ ___ _____ _ _____ _____-_____. »

espagnol déconverti : « por lo tanto nosotros confiamos en ti para animación y enrichment de sitio web este sitio web es tu sitio web. »

Nous remarquons que l'UW « website » se répète trois fois dans le graphe et que par suite le mot correspondant se répète trois fois en français, en russe, et en espagnol. Ce graphe est faux (il faut seulement deux nœuds « website »), mais illustre le problème, puisqu'on a a priori 6 façons de faire se correspondre les 3 nœuds « website » et les trois termes « site web ».

3.4.3 Description de l'algorithme

Nous proposons maintenant une procédure heuristique. Nous prenons toujours nos deux exemples pour illustrer ce problème de trouver la meilleure trajectoire. Nous illustrons d'abord les idées principales avec des figures et des exemples, et donnons les pseudo codes dans la section suivante.

Rappelons que nous avons du côté texte les structures de données de texte (M_i), le treillis LMS ($noeudtreillisetendu_i$) et l'ensemble des liaisons L_{12} . Du côté de l'arbre UNL, nous avons les structures de données de graphe UNL (G, $noeudgraphe_i$), de l'arbre UNL ($noeudarbreeteundu_i$), et l'ensemble des liaisons L_{34}.

Nous définissons d'abord une liaison lexicale :

liaison lexicale = (id_liaison, profil (=lexicale), noeudarbreetendu, noeudtreillisetendu).

Pour établir les liaisons lexicales, nous parcourons et comparons les lemmes anglais dans les neoudtreillisetendu et les lemmes français dans les noeudarbreetendu. Si nous trouvons le même lemme des deux côtés, nous créons une liaison lexicale entre ces deux nœuds liés.

Nous gardons une liste de liaisons lexicales et nous vérifions si tous les nœuds d'arbre ne sont liés qu'à un seul nœud de treillis et vice versa. Si oui, nous avons trouvé la meilleure trajectoire (par rapport à la situation courante). Toutes ces liaisons lexicales sont des « liaisons sûres ». Sinon, nous devons isoler tous les nœuds qui sont liés plusieurs fois et nous construirons un arbre de recherche et une liste d'attente pour chaque nœud isolé.

Voici une figure qui montre toutes les liaisons lexicales construites pour l'exemple (II). Les lignes pointillées sont des liaisons non sûres. Nous avons

liaison_sûre = $\{l_5, \ldots, l_{10}\}$.

nœudarbreetendu_lié_plus_une_fois = {lsea, 2sea}.

liaison_à_vérifier = $\{l_1, l_2, l_3, l_4\}$.

Nous cherchons d'abord une trajectoire contenant le plus possible de « liaisons sûres ». Ici, en utilisant ces liaisons sûres, nous pouvons construire sur le treillis une trajectoire provisoire qui contient les nœuds suivants :{débuttreillis _ 5Aral _ 9quatrième _ 11plus _ 12grand _ 14intérieur _ fintreillis} (ces nœuds sont en carré gras).

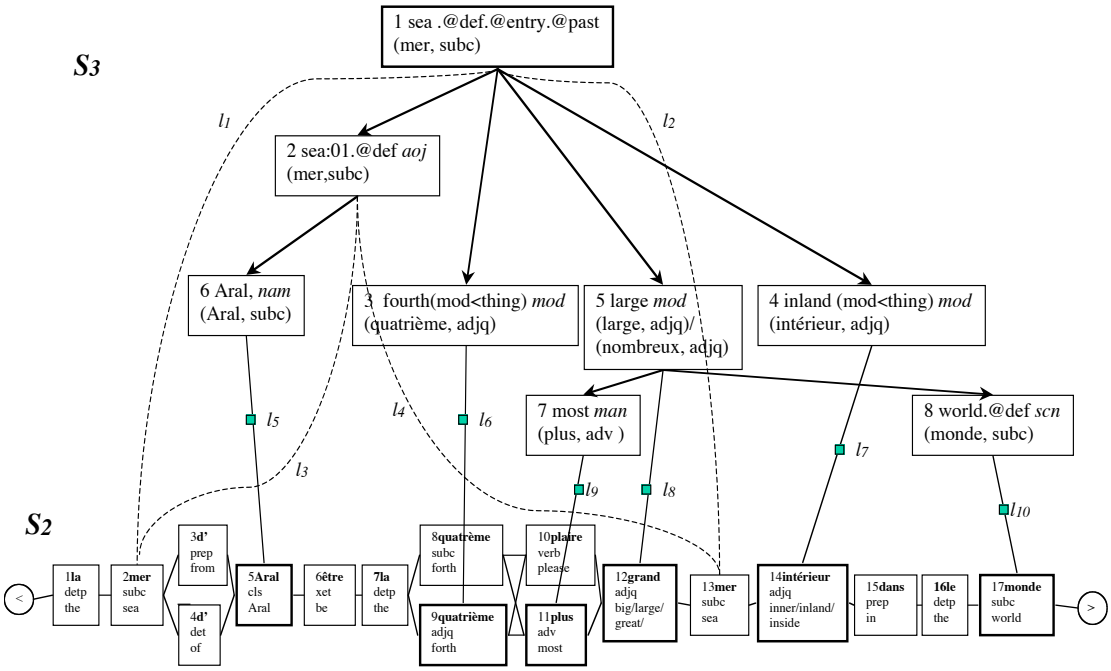


Fig. C-63 Trajectoires provisoires de l'exemple (II)

Nous construisons ensuite les listes d'attente pour les deux noeudarbreetendu « 1sea » et « 2sea ». La liste d'attente pour « 1sea » est [l_1, l_2,] et celle pour « 2sea » est [l_3, l_4]. Partant de la trajectoire provisoire ci-dessus, on ajoute la liaison l_1 dans la trajectoire et on met la liaison l_2 en attente. Puis on commence une autre liste d'attente, [l_3, l_4]. Cette fois-ci, on prend la liaison l_3. Quand on est au bout d'un chemin, on calcule la pénalité de croisement. Mais l_3 et l_1 sont exclusives l'une de l'autre, parce qu'elles sont liées au même noeudtreillisetendu « 2mer ». Donc, on quitte ce chemin et on retourne à l'état précédent dans l'arbre de recherche.

Nous aurons donc l'arbre de recherche suivant :

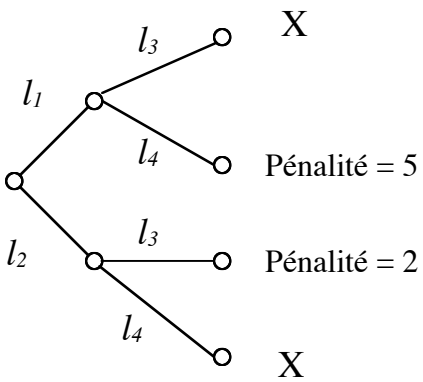


Fig. C-64 Arbre de recherche

La définition formelle d'un croisement sera donnée dans la section suivante, ainsi que la procédure de détection des croisements.

Voici les deux résultats de ce calcul de pénalité de croisement.

Le premier chemin est l_2 _l_3 (pénalité =2), et donne la correspondance illustrée par la Fig. C-65.

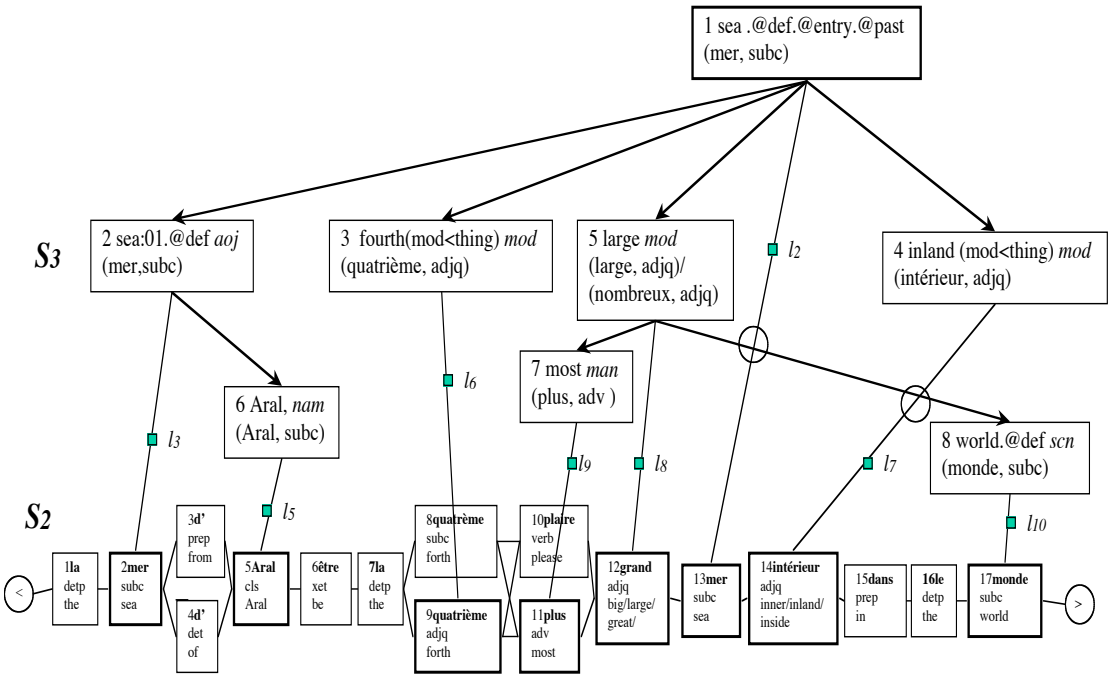


Fig. C-65 Liaisons lexicales (I), pénalité de croisement = 2

Le deuxième chemin est l_1_ l_4 (pénalité=5) et donne la correspondance illustrée par la Fig. C-66.

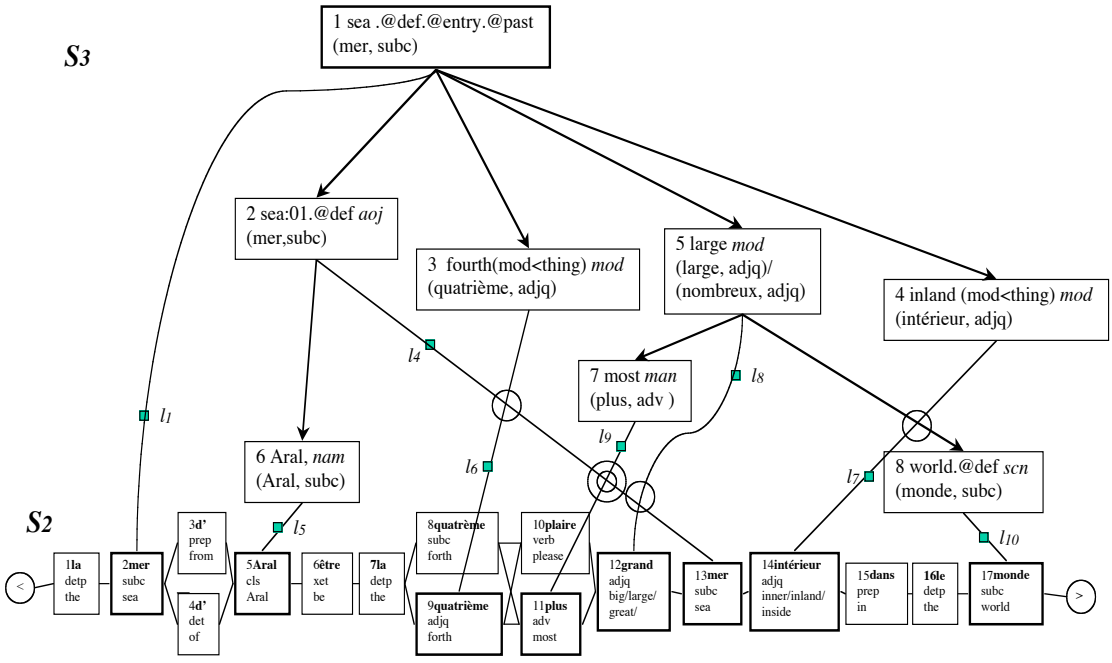


Fig. C-66 Liaisons lexicales (II), pénalité de croisement = 5

Selon notre algorithme, le croisement de l_4 et l_9 est plus important que les autres, il est donc compté deux fois. Cela donne une pénalité totale de 5, bien qu'il n'y en ait que quatre dans la figure.

La première trajectoire candidate sera choisie. On quitte la procédure de calcul de pénalité de croisement et on continue l'étape suivante en appliquant « enrichir_correspondance ».

Voici les liaisons lexicales de l'exemple (I) :

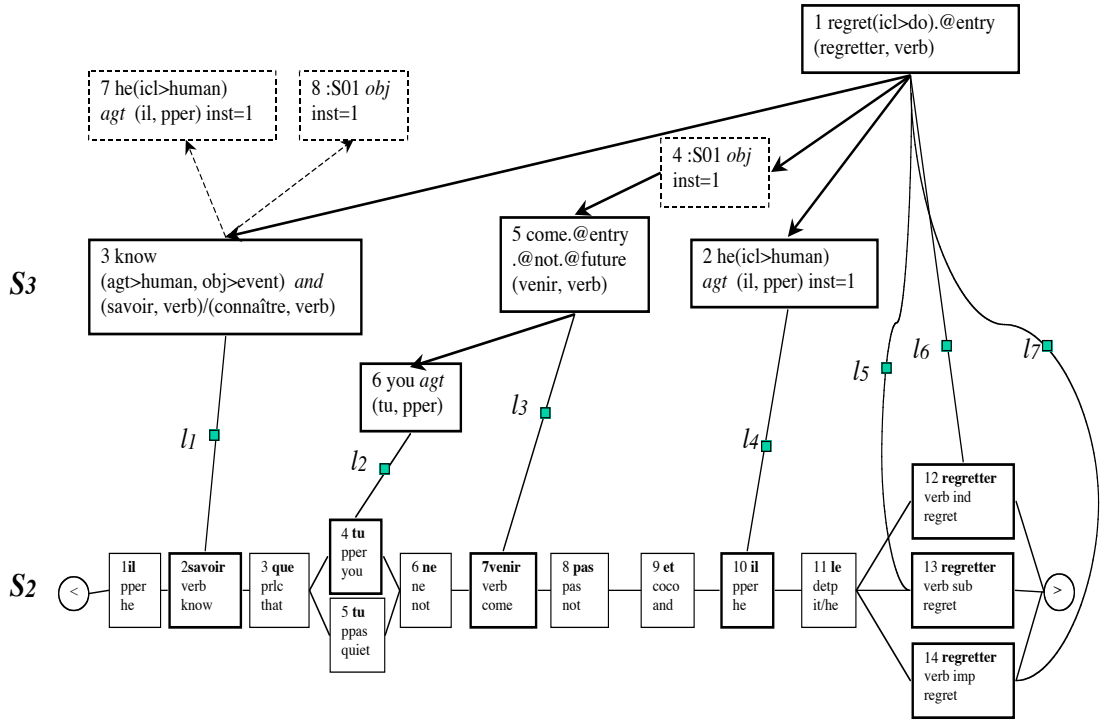


Fig. C-67 Trajectoires provisoires de l'exemple (I)

Pour ne pas compliquer le calcul, les nœuds clones et les pseudo-nœuds ne doivent pas participer à la procédure de la construction de liaison sûre. Par contre, on peut très bien les retrouver plus tard en suivant la structure d'arbre UNL si on veut les lier aux autres éléments du treillis.

Le problème de l'exemple (I) n'est pas le même que celui de l'exemple (II). On peut trouver trois trajectoires candidates et les scores de pénalité sont tous zéro (à cause du lemme « regretter » qui se répète trois fois avec différentes variables grammaticales).

Dans ce cas, on prend ces trois trajectoires candidates et on entre dans la prochaine étape « enrichir_correspondance ». On crée d'autres liaisons tentatives, et en même temps on calcule le poids de cette correspondance selon la sûreté attribuée à chaque liaison. On décidera la meilleure trajectoire selon son poids.

En fait, on verra plus tard que, même si à la fin de calcul on ne peut pas décider quel nœud parmi ces trois nœuds à choisir, cela n'empêche pas de construire la trajectoire et de lier correctement le mot « regrette » et le nœud du graphe « regret(icl>do).@entry ». On pourrait aussi proposer aux utilisateurs de choisir le bon groupe de variables (donc le nœud), ou simplement de fusionner ces nœuds (ici trois), en un « super nœud » avec trois ensembles de valeurs des variables.

La procédure enrichir_correspondance consiste à créer de nouvelles liaisons tentatives,

en utilisant des informations sur les catégories et les variables,

en se limitant à des liaisons compatibles avec l'ordre courant (n'ajoutant pas de croisement et/ou faisant intervenir des nœuds proches de nœuds déjà liés dans la correspondance courante),

en calculant le poids de cette correspondance enrichie.

S'il y a plusieurs trajectoires candidates, la trajectoire avec le poids le plus important sera choisie.

3.4.4 Structure de données et calcul possible

Avant de décrire la procédure complète, nous discutons d'abord la définition et la détection de croisement, puisqu'elle est un point essentiel pour décider la meilleure trajectoire. Puis, nous identifions les profils de liaison de L_{23}. Il existe beaucoup plus de profils dans L_{23} que dans L_{12} et L_{34}.

3.4.4.1 Définition et détection de croisement

Nous définissons maintenant une procédure pour détecter les croisements dans la correspondance arbre-chaîne et le calcul de pénalité.

Nous utilisons l'idée de SSTC [Boitet 88a] que nous avons présentée dans la section C.3.1. Nous enregistrons la sous-chaîne (STREE) correspondant à chaque sous-arbre de l'arbre UNL. Si un nœud correspond à un lemme qui est dans la sous-chaîne d'un autre sous-arbre auquel il n'appartient pas, il existe un croisement. Les deux figures suivantes (Fig. C-68 et Fig. C-69) décrivent ce phénomène.

Voici le premier cas. Supposons que nous avons réussi à établir quatre liaisons nœud – lemme après la consultation de dictionnaire. Na est la racine de cet arbre UNL et il a deux sous-arbres SA1 et SA2. La sous-chaîne (STREE) correspondant à SA1 est L1-Lc, et la sous-chaîne correspondante à SA2 est Lb-Li.

Nous constatons que le lemme correspondant (son SNODE, Lc) au nœud Nc est dans la sous-chaîne de SA2, donc il y a un croisement, de même pour Nb.

Si nous utilisons le nombre de croisements comme score de pénalité le score de pénalité de cette correspondance sera égal à 3 (les liaisons Na-La et Nb-Lb tombent dans le STREE de SA1 et la liaison Nc-Lc tombe dans le STREE de SA2).

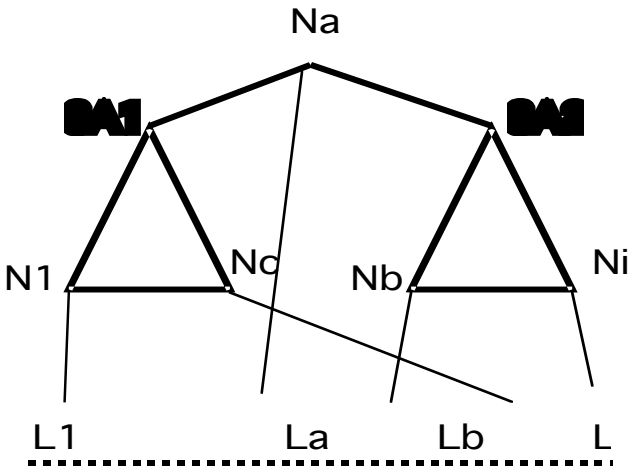


Fig. C-68 Croisement dans la correspondance arbre – chaîne (I)

Dans la figure suivante, le score de pénalité est 1.

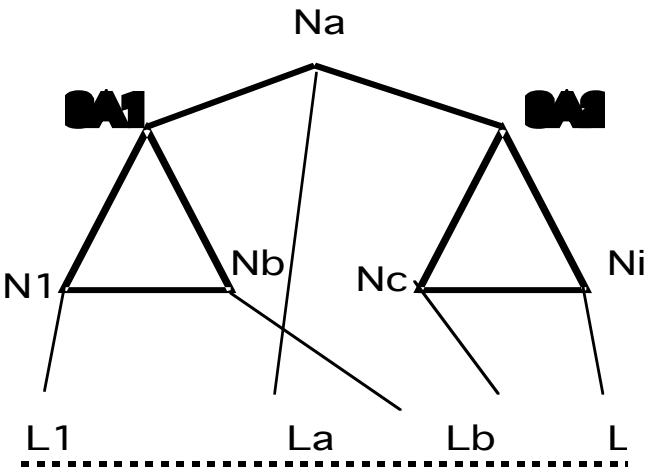


Fig. C-69 Croisement dans la correspondance arbre – chaîne (II)

Voici la procédure du calcul de la pénalité de croisement.

procédure calcul_croisement{
pour chaque nœud Ni dans l'arbre UNL avec son lemme correspondant Li {
pour chaque sous-arbre SAj {
si (Li tombe dans la sous-chaîne correspondant au sous-arbre SAj) et
(Li n'est pas un nœud dans le sous-arbre SAj)
{
pénalité de croisement ++
}
}
}
}

3.4.4.2 Profils de liaisons L_{23}

Avant de décrire la procédure « enrichir_correspondance », il faut présenter les « profils de liaison ». Le but de cette procédure est de créer le plus possible de liaisons pour que la correspondance L_{23} soit plus concrète et robuste.

Il y a plusieurs axes pour catégoriser les liaisons entre S_2 et S_3.

Selon l'ordre de construction :

sûre – liaison de base (toujours lexicale) construite dans un premier temps, en utilisant les dictionnaires.

secondaire –liaison construite à partir d'une ou de plusieurs liaisons sûres.

Selon la portée de la liaison, nous distinguons deux types :

verticale (intercouche) – les liaisons qui relient les deux couches.

horizontale (dépendance[20] dans le treillis) – selon la structure du graphe UNL, nous pouvons créer des liaisons qui spécifient la dépendance entre deux nœuds du treillis, par exemple, parce qu'on trouve un arc « sea→(inland, mod) » (ou « inland→(sea, xxmod) ») dans l'arbre, on peut créer une liaison dans le treillis entre « mer » et « intérieure » en spécifiant que « intérieure » est un modifiant de « mer ». Un autre exemple, c'est que nous pouvons lier facilement un nom et l'article défini ou indéfini devant lui.

Selon le niveau de d'analyse textuelle, nous distinguons trois types :

lexicale – entre un nœud d'arbre UNL et un nœud de treillis LMS.

grammaticale – liaison reliant un élément dans un nœud d'arbre UNL (par exemple, un attribut) ou un élément dans nœud de treillis LMS (par exemple, une variable grammaticale).

structurale – ces liaisons peuvent être créées pour marquer la projection d'un scope du graphe UNL sur un texte ou les autres correspondances structurales.

Nous reprenons le tableau des correspondances que nous avons donné dans la section C.2.2. et nous précisons les types de liaison entre le français (avec l'AMS PILAF) et le graphe UNL.

Graphe / sortie de PILAF	Graphe	sous-graphe/ scope	arc	relation	nœud	UW	restriction	attribut	type de liaison
Phrase	X								structurale
Sous-chaîne		X	X		X				structurale
Mot (forme)	X	X	X	X	X	X		X	lexicale
Lemme				X	X	X			lexicale
Catégorie							X		grammaticale
Variable grammaticale				X				X	grammaticale

Tableau C-9 Types de correspondance entre le français et le graphe UNL

Selon la présence des éléments des deux côtés, nous distinguons deux types :

binaire – les éléments liés aux deux couches sont présents.

unitaire – il manque l'élément dans l'arbre ou l'élément dans le treillis qui manque. Par exemple, basé sur la liaison lexicale « regrette – regret.@entry », on sait que le verbe est au temps présent, même si l'attribut .@present n'est pas dans le nœud de l'arbre.

Selon le nombre d'éléments liés à chaque couche, nous distinguons trois types :

[20] L'usage de cette liaison horizontale doit rester minimal dans la construction de L_{23}, car nous visons ici les calculs de liaisons entre graphe-treillis et nous souhaitons une méthode modulaire. Nous ne voulons pas créer des règles grammaticales. Mais on peut en ajouter après pour compléter la correspondance texte-graphe.

un _ un – la liaison relie un seul élément d'une couche à un seul de l'autre.

un _ plusieurs – la liaison relie un seul élément d'une couche et plusieurs dans l'autre.

plusieurs _ plusieurs – la liaison relie plusieurs élément dans chacune des couches.

Selon la sûreté de la liaison : nous pouvons aussi donner un poids à chaque liaison dans la table de compatibilité.

Pour marquer les profils des liaisons L_{23}, pendant le calcul, nous utilisons quatre profils : structural, lexical, grammatical, horizontal. Les autres types de liaison peuvent être créés après la construction d'une première correspondance pour compléter L_{23}.

3.4.4.3 Construction de liaisons lexicales

Nous abordons maintenant la structure de données et l'algorithme pour l'implémentation de la construction des liaisons entre l'arbre UNL et le treillis LMS (L_{23}).

Rappelons que nous avons du côté texte les structures de données de texte (M_i), le treillis LMS (noeudtreillisetendu$_i$) et l'ensemble des liaisons L_{12}. Du côté de l'arbre UNL, nous avons les structures de données de graphe UNL (G, noeudgraphe$_i$), l'arbre UNL (noeudarbreeteundu$_i$), et l'ensemble des liaisons L_{34}.

Noeudtreillisetendu=(id_noeudtr, lemtr, forme, tab_cat, tab_variable*, (lemmeanglais, catanglais)*,l_précédent, l_suivant)

id *lemme* *forme* *variables grammaticales* *anglicisation* *structure*

Noeudarbreetendu=(id_noeudar, lemar, tab_restriction, tab_attribut*, tab_rel, (lemmefrançais, catfrançais)*,npère, nfils)

id *Headword* *décoration* *francisation* *structure*

Fig. C-70 Structures des nœuds de treillis et d'arbre

Nous avons défini une liaison lexicale *liaison_lexicale* comme un quadruplet (id_liaison, profil(=lexical), noeudarbreetendu, noeudtreillisetendu).

L_{lex} est l'ensemble des *liaison_lexicale*.

Nous introduisons deux variables de liste pour enregistrer le nombre de liaisons créées sur noeudarbreetendu et sur noeudtreillisetendu. Initialement, les valeurs sont toutes nulles, la longueur est le nombre de nœuds.

no_liaison_noeudtreillisetendu= [0,0,0,…0] (dans exemple (II), la longueur de cette liste est égale à 17, car il y a 17 noeudtreillisetendu dans le treillis).

no_liaison_noeudarbreetendu= [0,0,0,..0] (dans exemple (II), la longueur de cette liste est égale à 8).

Nous aurons besoin d'un vecteur (de longueur variable) de booléens pour enregistrer les liaisons sûres : si une liaison est prise pour construire la trajectoire, sa valeur est

mise à un. Initialement, les valeurs de ce vecteur sont toutes nulles. La longueur de ce vecteur est égale au nombre de liaisons lexicales trouvées.

liste_liaison_lexicale= [0,0,0,..0]

<u>première étape : construction des liaisons lexicales</u>

procédure créer_liaison_lexicale
{
(pour chaque i dans noeudarbreetendu$_i$) {
(pour chaque j dans noeudtreillisetendu$_j$) {
si (il existe lemar$_i$ = lemmeanglais$_{jk}$) && (il existe lemtr$_j$ = lemmefrançais$_{ik}$)
{
//nous créons une liaison lexicale :
mettre liaison_lexicale$_{ij}$ dans L_{lex} ;
++no_de_liaison ;
++ no_liaison_noeudarbreetendu [i] ;
++ no_liaison_noeudtreillisetendu [j] ;

}
}}

// après cette procédure, on peut établir 10 liaisons lexicales, nous avons donc
// no_liaison_noeudarbreetendu = [2, 2, 1,1,1,1,1,1],
// no_liaison_noeudtreillisetendu = [0,2,0,0,1,0,0,1,0,0,1,1,2,1,0,0,1]
// liste_liaison_lexicale= [1,1,1,1,1,1,1,1,1,1] (longueur égale à 10)
// L_{lex} = {(l_1, 1sea, 2mer), (l_2, 1sea, 13mer), (l_3, 2sea, 2mer), (l_4, 2sea, 13mer),
// (l_5, 2 Aral, 5Aral), (l_6, 3fourth, 9quatrième), (l_7, 4inland, 14intérieur),
// (l_8, 6large, 12grand), (l_9, 7most, 11plus), (l_{10}, 8world, 17monde)}

procédure vérification_liaisons_lexicales {
meilleure_trajectoire_trouvée = 0 ;
si (tout no_liaison_noeudarbreetendu[i]<2) && (tout
no_liaison_noeudtreillisetendu[i]<2)
{ meilleure_trajectoire_trouvée = 1 ;
meilleure_trajectoire = {lemtr$_i$*} ;
meilleure_structure_arbre = {noeudarbreetendu$_i$*} ;
enrichir_correspondance }
sinon
{calcul_pénalité}
}

Si tous les nœuds de l'arbre UNL correspondent chacun à un seul nœud dans le treillis, le calcul s'arrête et la meilleure et d'ailleurs unique trajectoire a été trouvée. Sinon, il faut calculer la pénalité de croisement.

3.4.4.4 Calcul de pénalité de croisement

Ensuite, nous entrons dans la deuxième étape de la procédure.

deuxième étape : calcul de pénalité de croisement

procédure calcul_pénalité {
construire_liste_d'attente ;
construire_trajectoire_provisoire ;
construire_sous_arbre ;
construire_sous_chaîne ;
tant que liste_attente n'est pas vide {calcul_croisement ;}
choisir la pénalité minimale ;
meilleure_trajectoire_trouvée = 1 ;
enrichir_correspondance ;
}

procédure construire_liste_d'attente {
si (no_liaison_noeudarbreetendu[i]>1)
{construire liste_d'attente_noeudarbreetendu$_i$;
mettre toutes les liaisons lexicales de noeudarbreetendu$_i$ dans
liste_d'attente_noeudarbreetendu$_i$;
no_liaison_noeudarbreetendu [i] =0;
liste_liaison_lexicale [i] = 0;
}
si (no_liaison_noeudtreillisetendu[i]>1)
{construire liste_d'attente_noeudtreillisetendu$_i$;
mettre toutes les liaisons lexicales de noeudtreillisetendu$_i$ dans
liste_d'attente_noeudtreillisetendu$_i$;
no_liaison_noeudtreillisetendu [i] =0;
liste_liaison_lexicale [i] = 0;
}
}

procédure construire_projectoire_provisoire {
liaison_sûre = liste_liaison_lexicale;
projectoire_provisoire = tous les noeudtreillisetendu dans liste_liaison_lexicale;
}

procédure construire_sous_arbre {
(pour chaque i dans noeudarbre$_i$) {
(pour chaque noeudarbre$_j$ dans nfils$_j$)
{ sousarbre$_i$ = _ sousarbre$_j$ }}
}

Nous définissons ici les deux variables *s_node_noeudarbreetendu* et *s_tree_noeudarbreetendu*.
s_node_noeudarbreetendu est le noeudtreillisetendu correspondant dans la trajectoire provisoire, on peut le trouver par la liaison lexicale qui relie ces deux nœuds.

s_tree_noeudarbreetendu est un couple qui définit la limite de la projection du sous-arbre de ce nœud sur le trajectoire provisoire. Si ce nœud n'a pas de fils, s_tree_noeudarbreetendu = s_node_noeudarbreetendu.

procédure construire_sous_chaîne {
pour (chaque noeudarbreetendu$_i$)
{remplir (s_node_noeudarbreetendu$_i$, s_tree_noeudarbreetendu$_i$)}
}
procédure calcul_croisement {
pour (chaque s_node_noeudarbreetendu$_i$) {
pour (chaque s_tree_noeudarbreetendu$_j$) {
si (s_node_noeudarbreetnedu$_i$ tombe dans entre la limite de
s_tree_noeudarbreetendu$_j$) && (noeudarbreetendu$_i$ n'est pas dans le sous-
arbre de noeudarbreetendu$_j$)
{++ croisement }
}}
penalité_de_croisement= nombre de croisement ;
}

3.4.4.5 Enrichir la correspondance et calculer le poids

troisème étape : enrichir la correspondance et calculer le poids

On prend la (les) trajectoires avec le moins de croisements, on enrichit la correspondance, et on calcule son poids. S'il y a plus d'une trajectoire candidate, on prend la trajectoire avec le poids le plus important.

La procédure « enrichir_correspondance » consiste simplement à parcourir la table de compatibilité et on crée pour chaque élément les autres liaisons possibles.

procédure enrichir_correspondance {
pour chaque élément dans chaque noeudtreillisetendu {
parcourir la table de compatibilité et créer des nouvelles liaisons vers
noeudarbreetendu}
pour chaque élément dans chaque noeudarbreetendu{
parcourir la table de compatibilité et créer des nouvelles liaisons vers
noeudtreillisetendu}
calculer le poids total de cette correspondance ;
}

Voici la table de la compatibilité avec les poids pour chaque patron. Les poids ont été affectés de façon intuitive, suite à l'étude manuelle du corpus, et référant simplement notre « confiance » dans les différents types de liaison. Dans le futur, on pourrait les réévaluer automatiquement en fonction de la fréquence des différents types de liaison.

Poids d'une pénalité de croisement			10
Poids d'une liaison lexicale sûre			10
Poids d'une liaison lexicale secondaire			5
PILAF		**UNL**	**poids**
catégories			(*/5)
adv	Adverbe	(icl>how)	5

subc	substantif commun	(icl>thing)	5
adjq	Adjectif qualificatif	(mod<thing)/(aoj>thing)	4
verb	Verbe	(icl>do)/(icl>occur)/(icl>state)	5
detp	Déterminant-ponom	@def	3,5
ide	Indéfini	@indef	3,5
locp	Locution prépositionnelle	plc, tim	3
vet	Verbe être	aoj	3
xet/xav & ppas	Auxiliaire être/ Auxiliaire avoir & Participe passé	.@complete/.@past	4
ne pas	Négation ne & 2ème négation pas	.@not	5
variables			
imp	Impératif	.@imperative	4
fut	Futur	.@future	4
pre	Présent	.@present	3
imi	Imparfait de l'indicatif	.@past	3
cdl	Conditionnel	.@request/.@unreal	2
sub	Subjonctif		
plu	Pluriel	.@pl	5

Tableau C-10 Table de compatibilité

Voici une figure qui montre la correspondance enrichie après la procédure « enrichir_correspondance ». Nous distinguons quatre profils de liaison : une liaison lexicale sûre est représentée par un carré ; une liaison lexicale secondaire est représentée par un carré vide ; une liaison grammaticale est représentée par un cercle ; et une liaison horizontale est représentée par une croix vide.

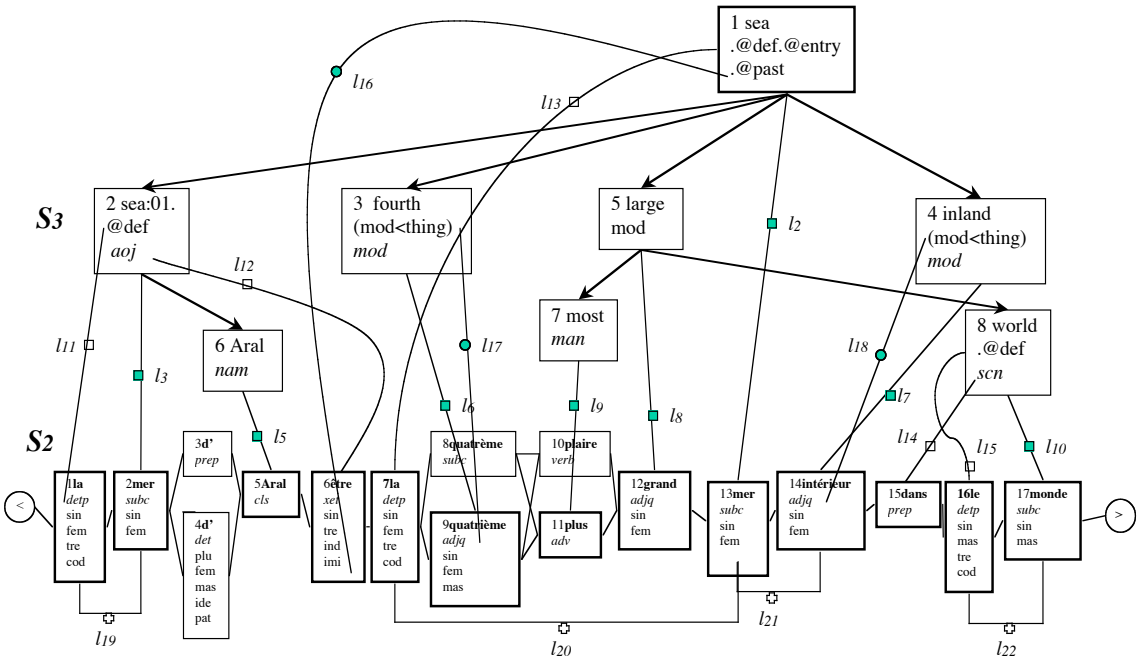


Fig. C-71 Correspondance enrichie

Nous avons donc les liaisons suivantes :

numéro	profil	élément arbre	élément treillis	poids	remarque
1	lexicale	1sea	2mer	10	sûre
2	lexicale	1sea	13mer	10	sûre
3	lexicale	2sea	2mer	10	sûre
4	lexicale	2sea	13mer	10	sûre
5	lexicale	2Aral	5Aral	10	sûre
6	lexicale	3fourth	9quatrième	10	sûre
7	lexicale	4inland	14intérieur	10	sûre
8	lexicale	6large	12grand	10	sûre
9	lexicale	7most	11plus	10	sûre
10	lexicale	8world	17monde	10	sûre
11	lexicale	@def	1la	5	secondaire
12	lexicale	aoj	6être	5	secondaire
13	lexicale	@def	7la	5	secondaire
14	lexicale	scn	15dans	5	secondaire
15	lexicale	@def	16le	5	secondaire
16	grammaticale	@past	imi (imparfait)	3	
17	grammaticale	(mod<thing)	adjq	4	
18	grammaticale	(mod<thing)	adjq	4	

numéro	profil	premier élément	deuxième élément	remarque
19	horizontale	1la	2mer	
20	horizontale	7la	13mer	
21	horizontale	14intérieur	13mer	
22	horizontale	16le	17monde	

Tableau C-11 Liste des liaisons trouvées

Voici enfin un diagramme résumant les étapes principales de notre algorithme.

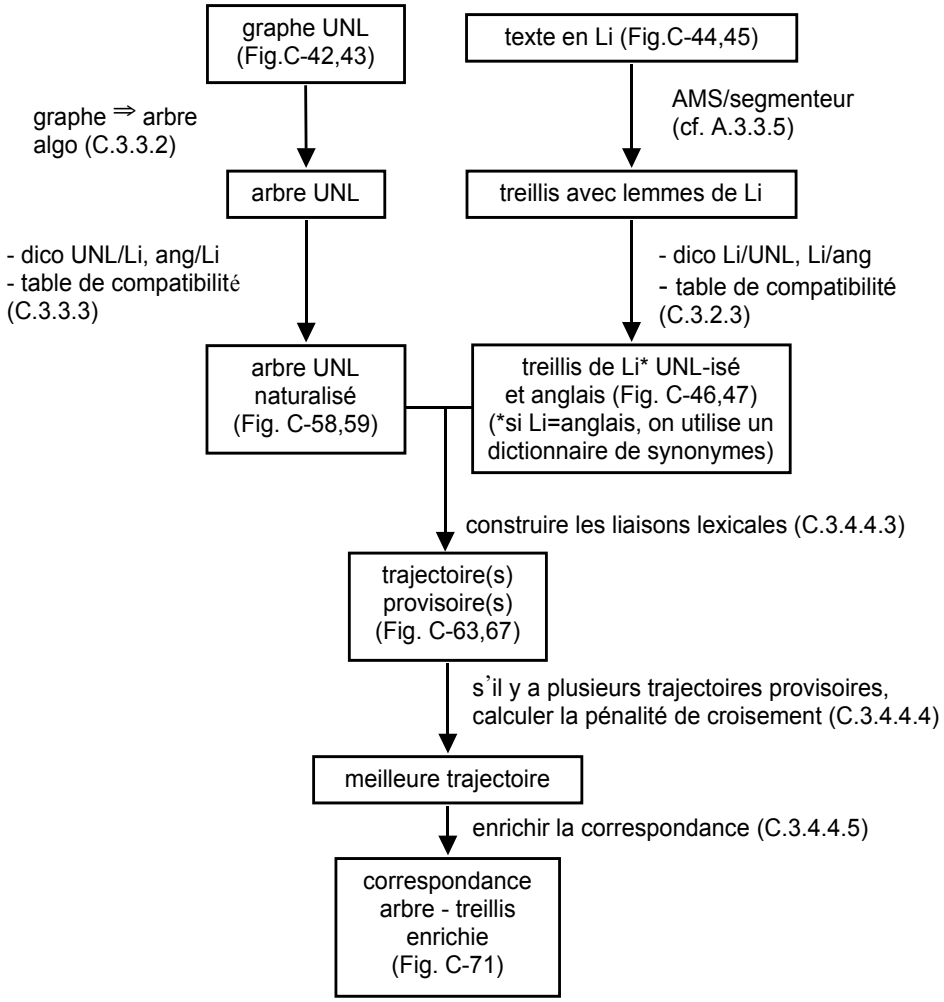


Fig. C-72 Procédure pour établir la correspondance texte - graphe UNL

D. Implémentation de la plate-forme SWIIVRE-UNL

Introduction

Nous avons décrit dans la partie précédente la procédure pour établir les correspondances entre le graphe UNL et le texte à travers un treillis AMS et un arbre UNL.

Nous présentons ici une plate-forme qui intègre les corpus et les modules UNL construits par les autres équipes, et qui peut servir comme environnement d'expérimentation d'UNL.

Cette plate-forme est le site web SWIIVRE-UNL (Site Web pour l'Initiation, l'Information, la Validation, la Recherche et l'Expérimentation d'UNL). Notre système de coédition s'intégrera dans ce serveur.

SWIIVRE-UNL est construit selon la méthodologie MultiCom [MultiCom99], et est accessible par Internet. Nous donnons le cahier des charges et les spécifications externes de ce site, et nous présentons les modules disponibles sur ce site, y compris la maquette de coédition.

Enfin, nous faisons un bilan de nos résultats.

1. Contexte et objectifs

1.1 Objectifs et motivations

1.1.1 Motivations

Il existe déjà un site UNL officiel. Ce site contient des informations sur le projet, et les documentations officielles. Mais il ne couvre pas tous les besoins. En particulier :

il ne permet pas d'essayer les outils et le graphe UNL.

l'environnement et les outils ni sont pas intégrés, mais dispersés sur tous les sites des LC (centres de langue).

pour les experts, il manque une plate-forme d'expérimentation.

le centre UNL a proposé quelques applications possibles d'UNL, mais il y a beaucoup d'autres possibilités et différents aspects qui existent mais le centre UNL ne mentionne pas.

nous ne trouvons pas sur ce site les corpus UNL qui sont nécessaires pour nos études sur la coédition.

il ne peut pas fournir les états courants des déconvertisseurs ou des autres modules essentiels pour l'utilisation d'UNL.

En bref, le site du centre UNL n'est pas destiné aux utilisateurs ordinaires, et ne le sera pas non plus dans l'avenir. En plus, le projet UNL existe déjà depuis un certain temps, mais personne n'a jamais essayé d'intégrer dans le site toutes les ressources et tous les résultats. De là vient notre l'idée de créer un autre site pour ça.

Notre site se veut indépendant du site UNL-C, des sites UNL-LC, du site FB2004, et des autres projets liés à UNL. Il est spécifique, et ne concurrence pas les autres sites UNL.

1.1.2 Cinq objectifs

Les cinq objectifs de cette plate-forme sont les suivants:

I (Information sur UNL) – accès au site du projet et informations spécifiques, par exemple, sur notre format UNL-xml et les outils associés, et aussi sur les développements du projet UNL. À partir de ce site, on aura une vue générale du projet UNL.

I (Initiation à UNL)– présentations pour non-spécialistes, exemples de graphes, conseils pour construire des graphes et initier les gens qui ne connaissent pas encore UNL.

V (Validation sur UNL) – validation de graphes UNL.

R (Recherche sur UNL) – cette plate-forme est le support expérimental de notre recherche sur UNL, et pourrait servir aussi à d'autres chercheurs.

E (Expérimentation avec UNL) – permet à des chercheurs et à des utilisateurs ordinaires de faire des expérimentations diverses.

1.2 Cahier des charges

1.2.1 Aspects généraux

Le site lui-même doit être multilingue – étant donné qu'UNL est un projet international et multilingue, il faut pouvoir dynamiquement changer la langue d'interface. Un objectif annexe est que le site soit « autodémonstratif », c'est-à-dire que la technique utilisée pour multilingualiser le site utilise elle-même UNL.

Utilisateurs visés – les utilisateurs visés sont de deux types, d'une part, des experts UNL qui cherchent des outils pour faciliter leur utilisation d'UNL ou connaître les développements et les recherches sur UNL, et d'autre part les débutants qui s'intéressent à UNL.

Implémentation du serveur – l'utilisateur devra pouvoir accéder au site par Internet. Le site sera donc implémenté comme un serveur web/http.

1.2.2 Ressources à récupérer et étapes de la récupération

L'intégration des outils existants est une étape principale de la construction de notre site.

Types de données à récupérer – il s'agit :

de liens vers d'autres sites : lien vers le centre UNL, et liens vers les autres serveurs UNL-LC et les autres sites discutant d'UNL.

de données « miroir » : spécifications du langage UNL, KB, et dictionnaires UNL-LN en format texte.

de données originales : corpus UNL complets ou incomplets de toutes formes, articles sur UNL en format texte, Word, ou PDF, transparents des présentations sur UNL, et logs de groupes de discussion sur UNL.

Types de modules à récupérer et à développer – il s'agit principalement des modules développés par les autres équipes locales UNL. Si le module peut être copié, nous récupérons une copie, sinon nous mettons un lien vers ce module. Nous avons produit nos propres modules seul et grâce à des stages d'étudiants. Les modules possibles sont : éditeur de graphe UNL, UNL viewer, UW Gate, tutoriel UNL, valideur UNL, déconvertisseurs, enconvertisseurs.

Étapes de la récupération – (1) d'abord il faut obtenir l'accord du propriétaire de la ressource ou du module. (2) Ensuite, selon les caractéristiques du module, soit on met un lien direct vers ce module si le module en question a une interface web, soit on adapte la configuration du module à l'environnement du site.

1.2.3 Descriptions des interactions et sorties

Types d'interaction – ce site est à la fois statique et dynamique : « statique » parce que le site fournit lui-même des informations et des données, « dynamique » parce

que le site peut aussi récupérer pour l'utilisateur des informations depuis d'autres sites ou déclencher des programmes et montrer leurs résultats.

Types des sorties à produire – les sorties statiques sont les informations sur UNL et sur les autres sites UNL, les articles et les spécifications UNL, les logs des états des déconvertisseurs, les logs de la génération automatique de graphes UNL, les corpus UNL, et les fichiers exemples du format UNL-xml.

Les sorties dynamiques sont les résultats de déconversion, de recherche d'information sur UNL, de consultation de dictionnaires UNL-LN, de transformation de fichiers UNL-html vers/depuis UNL-xml, et les courriels d'envoi de ces résultats à l'utilisateur s'il le souhaite.

Type d'utilisation du site – (1) consultation des informations, (2) téléchargement de données et ressources, (3) expérimentation de ressources.

Liens vers d'autres serveurs – (1) interrogations automatiques, (2) envois automatiques d'informations à d'autres serveurs, (3) réponses à des requêtes d'autres serveurs.

Contribution des utilisateurs – Cela pourra être réalisé quand nous aurons complètement intégré le module de coédition à notre site. L'utilisateur pourra corriger partiellement un document, ses améliorations étant alors ajoutées au document et gardées sur le serveur.

Échanges entre utilisateurs – Dans le cadre de ce travail de thèse, nous n'avons pas mis en œuvre cette fonctionnalité, car elle devrait plutôt être offerte par le site UNL principal. Elle se justifierait dans le cas d'un projet spécifique de développement coopératif, comme par exemple FB2004 (qui propose un tel forum) [FB2004], mais pas vraiment pour SWIIVRE-UNL.

1.3 Type de scénarios d'utilisation

Nous prévoyons les scénarios d'utilisation suivants pour la plate-forme SWIIVRE-UNL.

1.3.1 Accès au site

L'utilisateur peut accéder à notre site par Internet. Il lui suffit d'avoir un navigateur web. Certains modules sur le site peuvent aussi être accédés par une requête.

1.3.2 Choix de la langue de commande

Il est vrai qu'un document UNL-html est fortement multilingue, mais cela n'aide pas à construire un site web multilingue. En effet, dans un document UNL-html, les balises spécifiant les métadonnées (mise en page, lien, image, son, etc.) ne sont pas définies. Il nous faut une autre technique pour la multilinguisation de ce site.

L'architecture de ce site est multilingue, c'est-à-dire que les messages et les données textuelles des pages web sont dans des fichiers séparés. Pour l'instant, il n'y a que l'anglais, mais Hung [Hung 04] travaille à une multilinguisation de SWIIVRE-UNL : d'abord par TA/THAM « classiques », ensuite en ajoutant le langage UNL lui-même comme une autre version de langue.

1.3.3 Recherche des informations sur UNL

Supposons que quelqu'un cherche des informations sur UNL. Sur le site SWIIVRE-UNL, il trouvera les types d'informations suivants :

informations sur le site lui-même – page d'accueil montrant clairement les fonctionnalités disponibles sur le site, et les modifications récentes du site.

information sur le projet UNL – les prochaines conférences ou les prochains symposiums UNL, les nouveaux modules pour UNL, ou les nouvelles idées du centre UNL.

les dernières versions des spécifications d'UNL.

les états courants des modules UNL (notamment déconvertisseurs et enconvertisseurs) – l'utilisateur pourra obtenir un rapport sur les états courants de ces modules, et demander la mise à jour de ces données.

informations sur les recherches sur UNL – l'utilisateur pourra télécharger les articles et les transparents présentés dans les conférences ou l'introduction à UNL publiée sur Internet.

les liens vers les autres sites concernant UNL.

recherche par les moteurs – l'utilisateur pourra aussi cliquer sur des boutons pour lancer la recherche des informations sur UNL par son moteur de recherche préféré.

1.3.4 Initiation sur UNL

Si l'idée d'UNL plaît à l'utilisateur et s'il veut s'initier à l'UNL, il trouvera :

un tutoriel formé de quelques leçons simples sur UNL.

des exemples simples de graphes produits par des centres UNL-LN .

1.3.5 Essai et expérimentation de graphes UNL

Quand il aura une idée plus claire sur UNL, l'utilisateur sera prêt pour essayer la création de graphes UNL.

il pourra utiliser les éditeurs de graphes UNL disponibles sur le site.

il pourra accéder aux dictionnaires UNL-LN sur le site.

il pourra copier des exemples simples dans le tutoriel et essayer de les déconvertir. Sur le site, il trouvera une liste de déconvertisseurs.

il pourra éventuellement essayer les enconvertisseurs pour générer un graphe UNL à partir d'un texte dans telle ou telle langue. Il trouvera une liste des enconvertisseurs disponibles.

un vérificateur du graphe UNL l'aidera à produire le bon graphe.

le site l'aidera à apprécier le multilinguisme qu'UNL peut apporter. Toutes les déconversions ou enconversions peuvent être multilingues et synchrones.

1.3.6 Usage avancé

Jusqu'ici, nous avons décrit les scénarios d'utilisation par un utilisateur ordinaire. Après plusieurs essais et quelques études des spécifications UNL, on aura dans doute envie d'utiliser UNL pour faire des expériences plus intéressantes. Nous proposons pour l'instant les applications UNL suivantes.

XML-isation d'un document UNL.

coédition entre le graphe UNL et un texte.

En plus, d'autres informations pouvant intéresser les chercheurs sont aussi disponibles :

les corpus UNL.

les statistiques sur les serveurs UNL.

1.4 Réalisation

1.4.1 Méthodologie

Nous avons suivi la méthodologie de Multicom [Multicom 99] pour prototyper et évaluer SWIIVRE-UNL. Multicom est une plate-forme d'expérimentation et d'évaluation qui vise à répondre aux besoins suivants liés à la construction de systèmes interactifs :

Observer et diagnostiquer les usages,

Définir les besoins et le cahier des charges pour un système interactif à concevoir,

Écrire des scénarios d'utilisation,

Expérimenter ces scénarios sur des situations simulées ou réelles,

Valider le cahier des charges et spécifier le système,

Réaliser le système interactif,

Évaluer son utilisabilité.

Les domaines d'application sont multiples :

Rédaction Assistée par Ordinateur,

Traduction Assistée par Ordinateur,

Outils de Traitement Automatique des Langues Naturelles,

Interfaces graphiques et multimodales,

Dialogue homme-machine,

Commande vocale et multimédia,

Recherche d'informations multimédia,

Collecticiels, synergiciels (groupware, mediaspace),

Création multimédia (musique et graphique),

Conception de systèmes hypermédia,

Enseignement et formation assistés,

Multilinguisation des logiciels.

Notre site correspond à plusieurs domaines cités ici. Multicom a défini aussi le cycle de conception que nous avons utilisé dans le développement de notre site. Ce cycle de conception est le suivant :

A : Observer les usages des systèmes existants en vue d'un diagnostic d'usage,

B : Définir un cahier des charges initial CC(0) pour le nouveau produit ou système interactif,

C : Écrire et expérimenter les scénarios d'utilisation de ce nouveau système sur des situations simulées et des utilisateurs/sujets,

D : Affiner le cahier des charges CC(1) et spécifier le système,

E : Réaliser une maquette M(1),

F : Mettre la maquette en test (réel ou simulé),

G : Évaluer la signification d'usage et l'utilisabilité de cette maquette,

H : Corriger et modifier les spécifications,

I : Développer le système à l'aide d'une nouvelle maquette M(i),

J : Revenir à l'étape F tant que c'est nécessaire,

K : Évaluer le système final,

L : Réaliser un prototype ou une première série,

M : Le mettre sur le marché (dans notre cas, le mettre à disposition libre sur le web),

N : Former si besoin les utilisateurs ou les installateurs,

O : Analyser les retours du marché (dans notre cas, les retours des utilisateurs).

Notre prototypage a donc été mené en plusieurs étapes, de façon incrémentale. A chaque étape, nous avons ajouté des liens ou rendu de nouveaux outils disponibles.

1.4.2 Étape 0 : fonctionnalités statiques de base

Idée: intégration des modules UNL existants au site SWIIVRE

Le but de cette étape est de créer une plate-forme pour faciliter l'usage d'UNL et d'y intégrer les modules existants. C'est la version **CC(0)** du cahier des charges.

Il s'agissait d'abord de montrer les adresses utiles et les informations sur UNL. Ce site a plutôt servi de réservoir d'articles, et de liens relatifs à UNL.

Fonctions

Voici la page d'accueil de cette maquette. Le site a été ouvert le 20/03/2001. Il était très statique et n'avait pas beaucoup d'interaction avec l'utilisateur.

Ses fonctionnalités au début se limitaient à fournir :

l'introduction à UNL,

la collection des articles,

la collection des corpus,

les spécifications sous forme téléchargeable,

les liens vers des déconvertisseurs,

l'espace d'essai du graphe UNL.

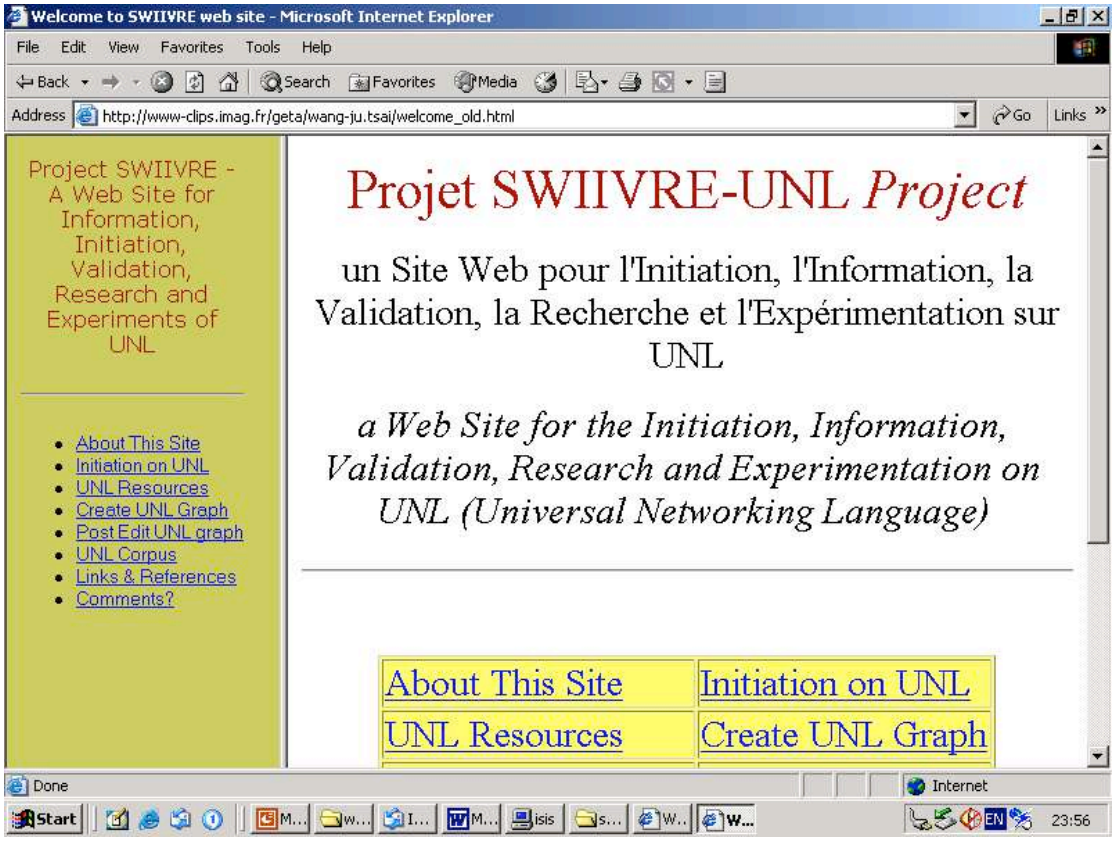


Fig. D-1 Interface du site SWIIVRE (version 1)

Évaluation

Ce version initiale du site offrait peu d'interactions avec les utilisateurs. Il a surtout servi de fournisseur d'informations et d'hyperliens à la communauté UNL.

Malgré quelques publicités dans les articles et les conférences internationales, le nombre d'utilisateurs est resté limité. Nous avons été contactés de temps en temps pour les informations sur UNL et nous avons toujours donné l'adresse du site SWIIVRE-UNL, mais nous n'avons malheureusement jamais eu de remarque d'utilisateur du site.

1.4.3 Étape I déconversion multilingue, éditeur UNL de base

Nous voulions que ce site ne soit pas seulement un réservoir de liens et d'articles. En plus, nous ne pouvions pas attendre que le centre UNL fabrique tous les modules dont nous aurions besoin. Avec des stagiaires, nous avons alors écrit quelques modules que nous pensions importants pour la communauté UNL, par exemple, un déconvertisseur qui montre plusieurs déconversions en parallèle et un vrai éditeur de graphe UNL qui peut aider à créer le graphe UNL, sachant que la plupart des enconvertisseurs n'étaient pas encore disponibles.

Nous avions déjà pensé à XML-iser un document UNL, donc cet éditeur de graphes UNL devait avoir la capacité de produire un document UNL-xml.

Le but de cette étape a donc été d'ajouter de nouveaux modules, de rendre le site plus dynamique, d'attirer l'intérêt des utilisateurs, et de mieux présenter la situation actuelle d'UNL.

Nous avons pour cela produit une nouvelle version du cahier des charges, celle que nous avons vue dans la section D.1.2 et que nous appelons aussi **CC(1).**

Fonctions

Nous avons ajouté plusieurs modules après deux stages réalisés pendant 6 mois :

Stage TER de Preedarat Jitkue (janvier 2001-avril 2001) : nous avons créé un déconvertisseur synchrone multilingue sur le serveur, qui peut envoyer un graphe UNL aux déconvertisseurs choisis par l'utilisateur et montrer les résultats en parallèle.

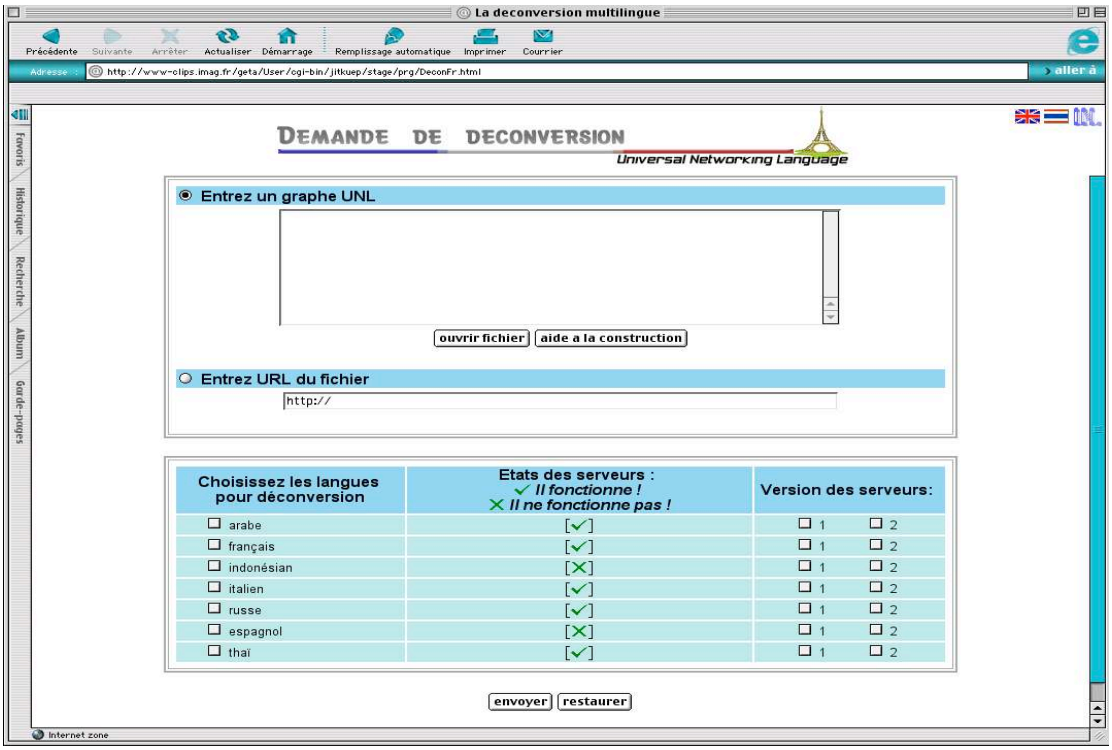


Fig. D-2 Déconvertisseur multilingue synchrone

Stage de maîtrise de Preedarat Jitkue (mai 2001 - août 2001) : nous avons créé un éditeur UNL de base qui permet aux utilisateurs de manipuler les représentations graphique et textuelle du graphe UNL.

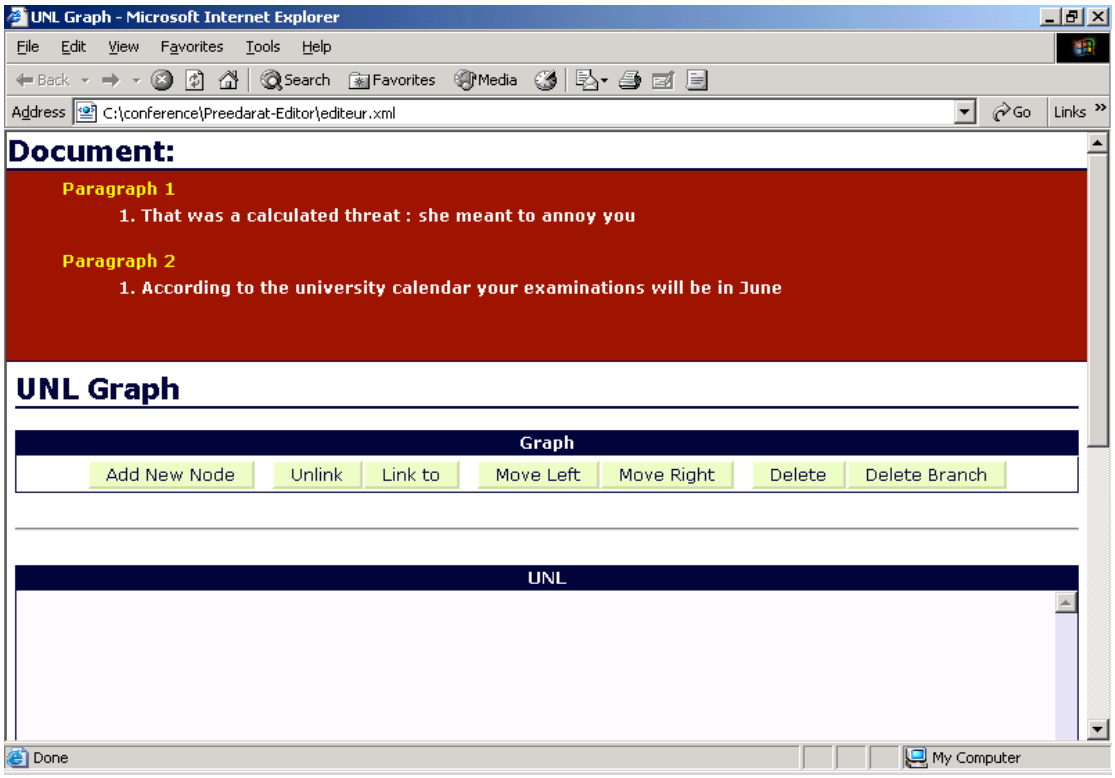


Fig. D-3 Interface de l'éditeur UNL de base

En bref, les modules ajoutés dans cette étape peuvent être catégorisés selon quatre des cinq objectifs de notre site (plus « interface » pour l'embellissement du site) :

interface - compteur de visites.

information - robot pour détecter les états des déconvertisseurs, le moteur de recherche pour les informations UNL sur Internet.

initiation - consultation des dictionnaires UNL-russe et UNL-italien en ligne.

recherche – visualiseur UNL-xml basique, et statistiques de déconvertisseurs.

expérimentation - génération automatique de graphes UNL (pour tester la limite et la couverture d'UNL), mais pas encore de documents UNL.

1.4.4 Étape II□ première réalisation de la maquette de coédition

La conception du site ayant été précisée, nous pouvions commencer la conception de la maquette de coédition et le rangement des corpus, pour observer les patrons de correspondance entre le graphe UNL et LN.

Le but de cette étape a donc été de nettoyer les corpus UNL et de les transformer en format UNL-xml, puis de réaliser une première maquette de coédition.

Nous avons aussi travaillé avec des stagiaires pour réaliser cette maquette spécifique.

Fonctions

Deux stages réalisés en six mois ont permis de créer une maquette de coédition en Java et en PHP (l'environnement de programmation Java utilisé, JBuilder, limite l'utilisation de la maqutte à Windows).

Projet DESS DCISS de Stéphane Helme, Delphine Bernhard, et Sandra Echinard : nous avons réalisé une première version de la maquette de post-édition partagée par coédition utilisant UNL. Dans cette maquette, nous avons écrit un programme en PHP qui marche sur un serveur local sur PC. Ce programme peut lire et afficher un fichier UNL-xml en appelant une XSLT. L'utilisateur peut choisir la partie du texte qu'il veut corriger. Ensuite une applet se lance pour la procédure de coédition du texte choisi.

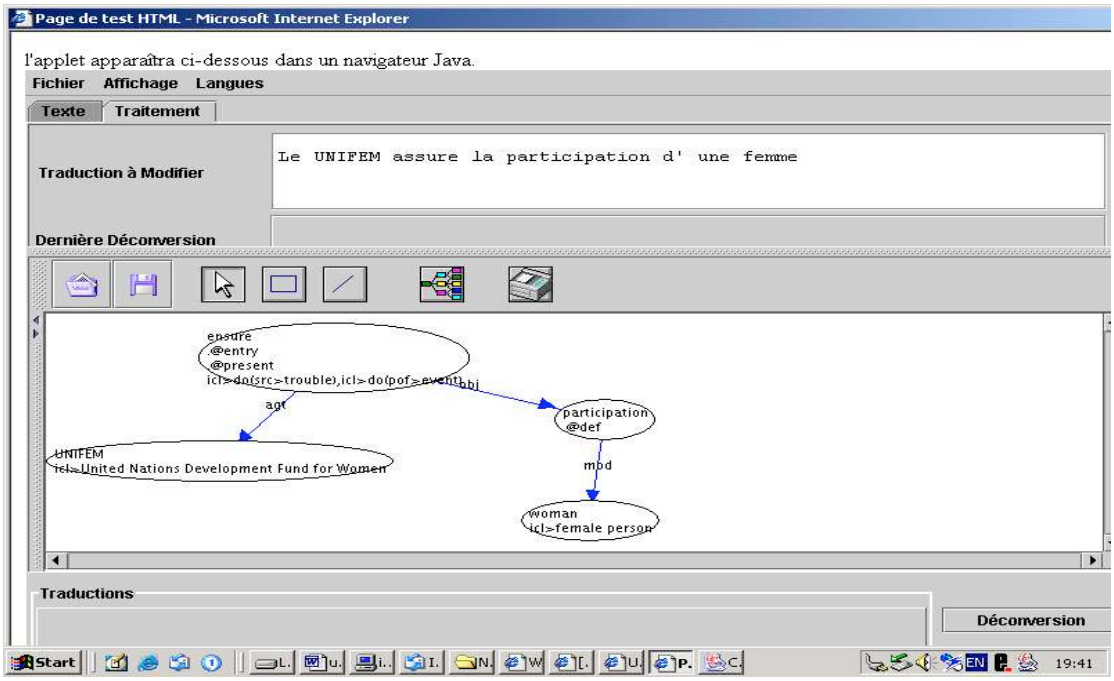


Fig. D-4 Applet de coédition

Stage d'été de DESS DCISS de Stéphane Helme : nous avons amélioré la maquette précédente et l'avons connectée au déconvertisseur français pour pouvoir demander la déconversion. La maquette a inclus aussi un dictionnaire UNL-français ainsi que PILAF, un AMS du français disponible sur le site du GETA. L'interface n'a pas été changée.

Nous reviendrons sur les détails de cette maquette dans la section D.2.2.

Nous avons changé la page d'accueil et la mise en page du site. Voici la nouvelle page d'accueil.

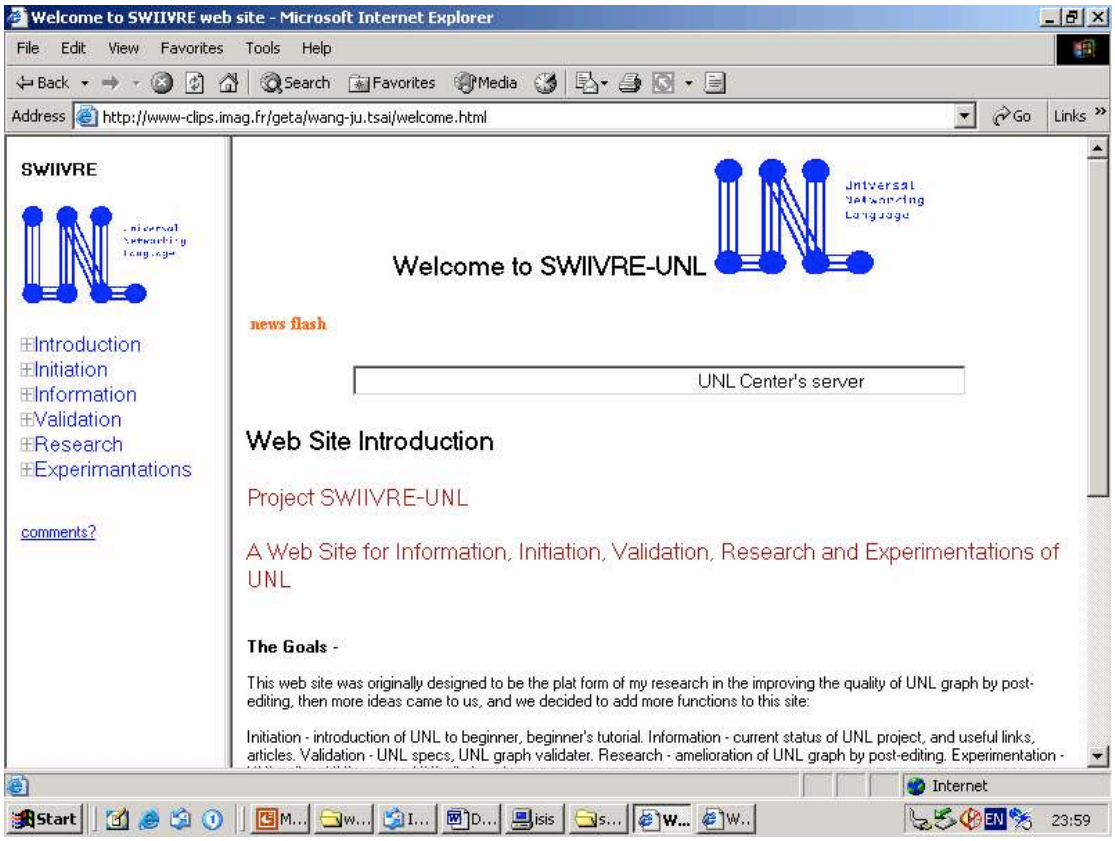


Fig. D-5 Page d'accueil de SWIIVRE-UNL (version 2)

Dans cette nouvelle interface, nous avons reconçu les catégories présentées à gauche pour qu'elles correspondent mieux aux cinq buts (I.I.V.R.E.) de SWIIVRE-UNL. Nous avons mis en tête les modules que nous pensions les plus intéressants. Les modules sont maintenant mieux rangés sous la catégorie à laquelle ils appartiennent.

Nous avons aussi mieux séparé les fonctionnalités statiques et dynamiques : quand l'utilisateur clique sur un bouton d'un programme, une nouvelle fenêtre s'ouvre, ainsi il peut continuer à parcourir le site sans être interrompu.

En bref, les changements sont :

> interface – une nouvelle interface et une nouvelle page d'accueil.
>
> validation – lien vers le « UNL verifier » du centre UNL.
>
> recherche - création de cette maquette de coédition, ajout des corpus XML-isés, du visualisateur UNL-xml, et d'un programme en Perl qui peut transformer la forme UNL-html en UNL-xml.

1.4.5 Étape III coopération avec « La main à la pâte »

Après UNL-2002 (Goa, Inde), nous avons décidé de tester l'utilisabilité d'UNL dans un cas réel, dans le cadre d'une coopération avec l'association « La main à la pâte ». Cela nous a mené à une nouvelle étape.

Pourquoi fallait-il une nouvelle étape de SWIIVRE-UNL ? L'idée était que, dans l'avenir, on puisse intégrer, dans un site quelconque, les « fonctionnalités UNL » :

remplacement d'un document monolingue par un document multilingue.

visualisation de ce document (sous un navigateur quelconque) dans toute langue disponible.

amélioration par coédition.

De là viennent les 3 sous-étapes suivantes :

(1) ajout de ces 3 fonctionnalités au site SWIIVRE-UNL (prototypage et expérimentation).

(2) consolidation pour implémentation sur un site quelconque.

(3) implémentation sur le site de « La main à la pâte » et expérimentation.

Les sous-étapes (1) et (2) sont détaillées ci-dessous, mais pas l'étape 3. Disons simplement que la sous-étape (3) est réalisée en transformant SWIIVRE-UNL en une « passerelle ».

Fonctions

Nous avons coopéré avec « La main à la pâte » pour tester la procédure de coédition.

« La main à la pâte » est une association (loi 1901) qui est responsable d'un site web international et multilingue de même nom, permettant à des enseignants de 7 pays de partager leurs méthodes et matériaux concernant la pédagogie des sciences dans le primaire.

Il s'agissait donc d'étudier comment intégrer les fonctionnalités UNL à ce site. Nous avons reçu un article court de 10 phrases, à partir duquel nous avons produit le graphe UNL manuellement, et l'avons déconverti automatiquement en 3 langues : russe, espagnol et italien.

La mise à jour des déconvertisseurs a commencé, mais nous devions aussi améliorer la maquette pour qu'elle puisse prendre cet article et compléter la procédure de coédition.

Dans ce cadre, Stéphane Helme a ajouté une fonctionnalité à la maquette pour qu'on puisse ajouter des patrons de correspondance.

Il a aussi créé un éditeur de graphes UNL, qui accède à un dictionnaire anglais-français et à la KB. Cet éditeur est écrit en Java pour la portabilité. Il a été conçu pour traiter les scopes (sous-graphes repliables) et la représentation graphique des graphes UNL. Nous l'appelons « éditeur UNL graphique ».

Voici une image de cet éditeur :

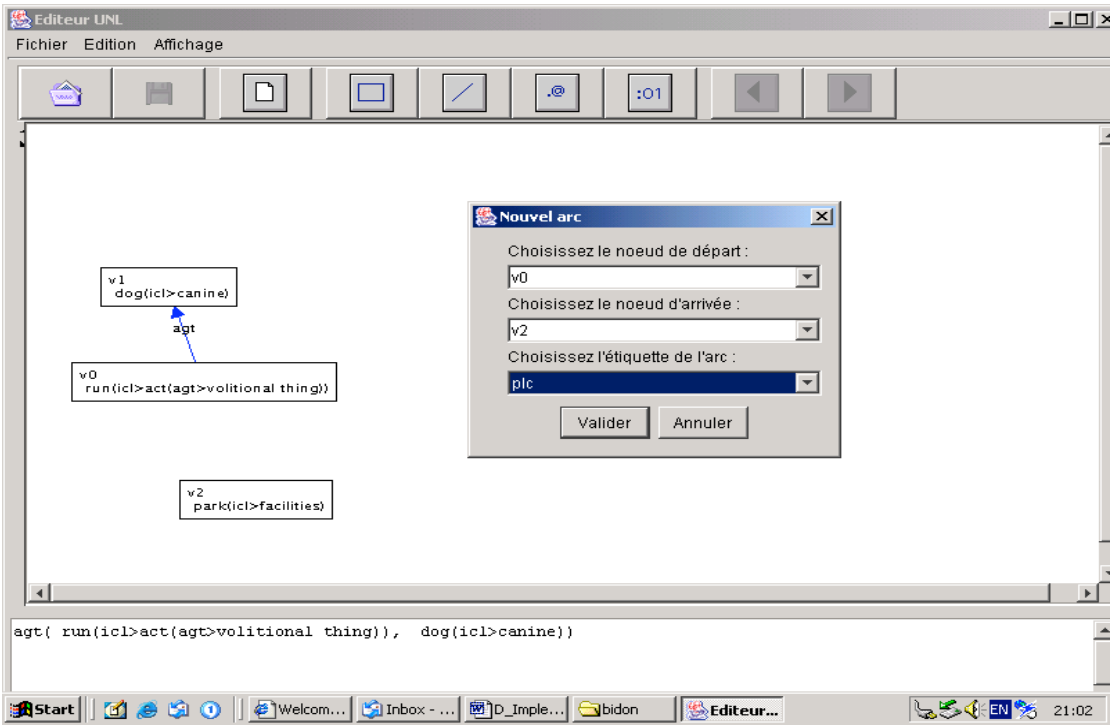


Fig. D-6 Editeur UNL graphique

1.5 État courant du site SWIIVRE-UNL (version 3)

Ce qui suit concerne l'état courant du site SWIIVRE-UNL fin 2003.

Fréquence de visite : il a été visité 920 fois en 2 ans (20/02/2002-29/02/2004). Nous ne disposons pas l'élément permettant de distinguer les différents visiteurs.

Fonctionnalités: (nous les classons en six catégories selon l'endroit où elles se trouvent sur le site). Ce sont des procédures actives et elles peuvent concerner un ou plusieurs objets (voir leur liste plus loin).

Six catégories :

Introduction – UNL news flash.

Initiation – téléchargement des spécifications UNL, consultation des information sur le site, édition du graphe UNL.

Information – détection des serveurs, liens vers les autres sites UNL, recherche des information UNL sur Internet, téléchargement d'articles sur UNL, envoi automatique des rapports d'état des serveurs.

Validation – lien vers « UNL verifier » du centre UNL.

Recherche – XML-lisation d'un fichier UNL-html, applications liées au format UNL-xml, génération aléatoire d'un graphe UNL, coédition d'un graphe UNL et d'une phrase.

Expérimentation – déconvertisseurs, enconvertisseurs, consultation de dictionnaires UNL-LN, éditeur graphique UNL, maquette de coédition.

Objets: ce sont des données de différentes formes.

UNL – fichiers de graphes UNL, exemples de graphes UNL.

URL – liens vers d'autres sites et vers d'autres modules.

UNL(-html) – corpus de graphes UNL.

HTML – page web, tutoriel UNL.

XML – fichiers UNL-xml, corpus UNL-xml.

Word (doc ou rtf), pdf, PowerPoint – articles, transparents, spécifications d'UNL.

texte – log des statistiques des serveurs, log du graphe UNL aléatoire, dictionnaire UNL-LN.

Le tableau suivant résume l'état courant du site SWIIVRE et les objectifs atteints. Pour chaque objectif atteint, nous marquons une lettre qui représente cet objectif :

I pour Information,

I' pour Initiation,

V pour Validation,

R pour Recherche,

E pour Expérimentation.

	Fichier HTML	Fichier UNL	Fichier XML	liens	corpus	dico	articles	spécs	tutoriel	log de déco	log aléatoire
consultation	I I'	I I' R	R		I I' R E	I I' R E	I I' R	I I' R V	I' E	I R E	I R E
télécharge-ment		I I' R	R		R	R	I I' R	I I' R V	I'	R	R
recherche d'information	I I' R			I I' R			I I' R				
liens vers autres sites	I I' R			I I' R			I I' R				
édition d'un graphe UNL		I' R E	R E								
validation du graphe UNL		V	V								
déconversion UNL-LN		I' R E	R E		R E						
déconversion multilingue		I' R E	R E		R E						
enconversion		I' R E	R E		R E						
envoi de rapports										I R E	I R E
coédition			R E								
détection de serveurs				I I' R E						I I' R E	
applications XML			R E								
génération aléatoire											R E

Tableau D-1 Fonctionnalités du site web SWIIVRE

2. Implémentation

Nous présentons ici les modules disponibles sur le site SWIIVRE-UNL, et nous discutons les algorithmes que nous avons exploités et les problèmes que nous avons rencontrés pendant l'implémentation. Nous décrivons la maquette de coédition dans une section séparée, parce qu'elle n'est pas liée au site lui-même.

2.1 Modules sur le site SWIIVRE

Il y a actuellement 7 modules sur le site SWIIVRE-UNL.

2.1.1 Détection de l'état des déconvertisseurs

Ce programme permet à l'utilisateur de savoir si un ou plusieurs déconvertisseurs sont actif(s) ou non. Avant d'envoyer le graphe UNL vers un déconvertisseur, il est peut-être sage de tester l'état de déconvertisseur. Le moyen de test est simple : l'utilisateur clique sur un bouton et notre serveur envoie le graphe suivant au déconvertisseur spécifié par l'utilisateur :

```
[S:1]
aoj(work(icl>occur).@entry, i)
tim(work(icl>occur).@entry, today)
[/S]
```

Si, par exemple, le déconvertisseur français fonctionne, il renverra la date, l'heure et le résultat de déconversion à l'utilisateur dans une autre fenêtre. En français, on obtient la phrase « je fonctionne aujourd'hui ». Pour l'instant, nous ne pouvons tester que quatre déconvertisseurs (français, russe, italien, espagnol), parce que ce sont les seuls qui sont ouverts au public. Voici l'image de ce module et la réponse du déconvertisseur russe dans la fenêtre plus petite.

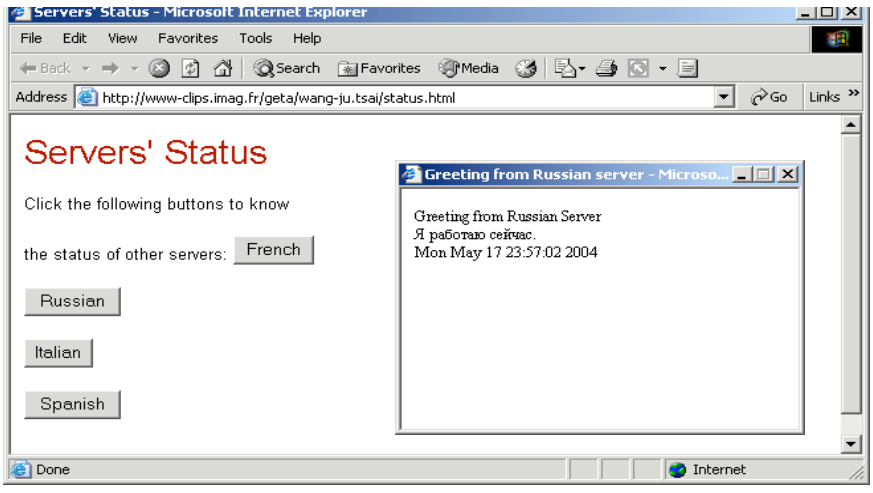


Fig. D-7 Tester les états des déconvertisseurs

Pour attirer des utilisateurs, la stabilité des déconvertisseurs et des enconvertisseurs est indispensable. Nous avons écrit un cronjob (tâche exécutée automatiquement par le système à une heure spécifiée) sous forme d'un script Unix qui envoie systématiquement deux fois par jour ce graphe vers quatre déconvertisseurs (français, russe, italien, espagnol), et envoie chaque jour le résultat du jour aux personnes

concernées par courriel. A la fin du mois, l'analyse du résultat est faite. Elle consiste simplement à compter le nombre de bonnes réponses et de messages d'erreur. Nous gardons les résultats depuis mars 2002 et on peut aussi les trouver sur le site.

Voici le rapport des états courants des serveurs, envoyé automatiquement tous les jours :

```
Tue Mar  2 00:00:00 MET 2004
   [D] [S] Je fonctionne aujourd'hui . [/S] [/D]
Tue Mar  2 00:15:00 MET 2004
_ _______ _______.
Tue Mar  2 00:30:01 MET 2004
[S:1]
yo trabajo hoy.
[/S]
Tue Mar  2 00:45:00 MET 2004
[S:1]
[S:1]io funzion oggi .
 [/S]
```

Voici une ligne de ce cronjob qui contacte le déconvertisseur français tous les jours à 00 :01 et à 12 :01 et sauvegarde le résultat de déconversion dans le fichier « todaylog » :

```
1 0,12 * * * /user/local/bin/lynx –post_data
http://gohan.imag.fr:8002/Deco.po <
/services/HtmlDocs1/clips24042001/geta/User/wang-
ju.tsai/decolog/unlfrdata
>>/services/HtmlDocs1/clips24042001/geta/User/wang-
ju.tsai/decolog/todaylog
```

Il faut noter que chaque déconvertisseur a ses particularités. Il faudrait bien sûr normaliser. Par exemple, quelques-uns n'acceptent pas de « retour chariot » dans un graphe UNL, donc il faut écrire séparément les données (y compris le mot de passe et le nom d'usager) à envoyer pour chaque déconvertisseur.

Voici les statistiques qu'on peut trouver sur le site SWIIVRE-UNL :

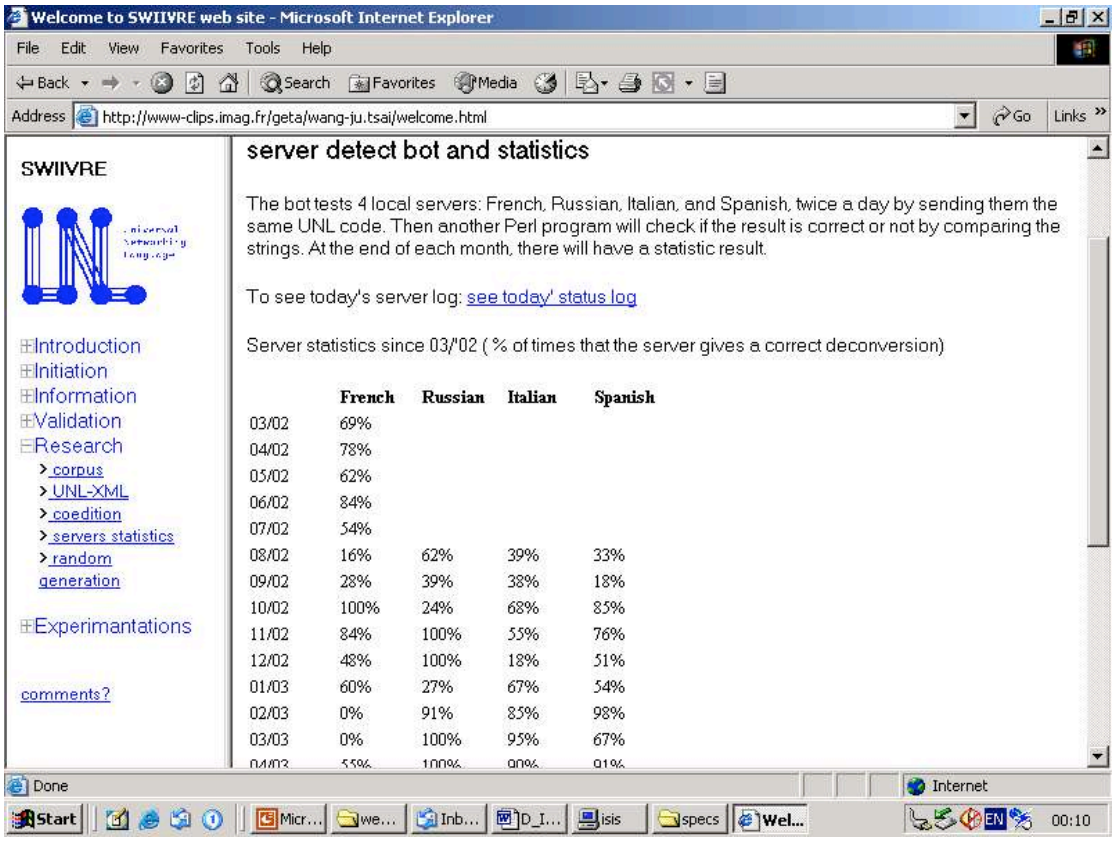


Fig. D-8 Statistiques sur les déconvertisseurs

La plupart des déconvertisseurs n'acceptent encore qu'un seul graphe à la fois. Pour obtenir le résultat de déconversion d'un article en UNL, nous avons écrit un programme en Perl qui découpe un document UNL en graphes UNL, appelle les déconvertisseurs choisis sur chaque graphe, et insère les résultats des déconversions dans le document UNL.

2.1.2 Test d'un graphe UNL aléatoire

Le script que nous avons présenté génère chaque jour aussi un graphe UNL simple de la forme suivante :

```
[S:1]
relation1 (UW1.@entry.@attribut1, UW2.@attribut2)
relation2 (UW1.@entry.@attribut1, UW3.@attribut3)
[/S]
```

Tous les UW, attributs et relations sont générés aléatoirement de façon à respecter si possible les contraintes exprimées dans les spécifications du langage UNL. Puis ce graphe est envoyé au déconvertisseur français, et le résultat est gardé dans le log des graphes UNL aléatoires. L'intérêt de ces tests est que nous pouvons tester la couverture et la robustesse des déconvertisseurs et en particulier la réponse du déconvertisseur français à des graphes UNL non conventionnels, et parfois même illégaux.

La génération automatique de corpus concrets plus ou moins annotés ainsi que de corpus formés de représentations abstraites IF ou UNL est une partie du sujet de la

thèse de Youcef Bey [Bey 03]. Nous remplacerons notre génération aléatoire, trop primitive, par une génération aléatoire plus élaborée dès qu'elle sera disponible.

Voici un extrait de ce log :

```
Sun Aug 24 17:55:00 MEST 2003
[S:1]
pof(work.@entry.@possibility,subject.@pl)
tmf(work.@entry.@possibility,understand(icl>action).@double_qu
otation)
[/S]

   trvail des sujets depuis une "compréhension " .

Fri Nov 14 17:55:01 MEST 2003
[S:1]
obj(make up(icl>do).@entry.@ability , mistrust.@interrogative)
tmt(make up(icl>do).@entry.@ability ,
maintain(icl>action).@obligation)
[/S]

   constitue une defiance à un maintien ?
```

Dans le premier exemple, « pof » veut dire « faire partie de » et « tmf » veut dire « fin de temps d'une action ». La restriction de « work » « .@possibility » ne peut pas être exprimée dans un groupe nominal de ce type (mais cela irait si « pof » était remplacé par « agt »). En bref, le graphe est trop petit pour exprimer « .@possibility ».

Dans le deuxième exemple, l'agent est absent, et donc la restriction « .@ability » du verbe ne peut pas non plus être exprimée.

En fait, la plupart des graphes produits sont refusés par le déconvertisseur. Les raisons sont deux :

> Les spécifications du langage UNL ont été modifiées plusieurs fois, les relations utilisées par le déconvertisseur français ont donc changé, mais nous n'avons pas intégré ces modifications dans les dernières ressources de ce générateur aléatoire.
>
> Le graphe aléatoire est petit, donc, certaines relations (de conjonction) ou restrictions ne s'appliquent pas.

Cette expérience nous est encore nouvelle. Soit on limite encore les applications des relations et des restrictions et on se contente sur la génération de petite phrase, soit on doit agrandir la taille du graphe aléatoire.

2.1.3 Editeur UNL de base et éditeur UNL graphique

Au cours du développement du projet UNL, plusieurs éditeurs de graphes UNL ont été créés (réf. Section B.2.2.3). Les stagiaires que nous avons encadrés ont ainsi créé deux éditeurs de graphes UNL.

Voici l'interface de **l'éditeur UNL de base**, réalisé par Preedarat Jitkue pour son stage de maîtrise durant l'été 2001 [Jitkue 01].

Cet éditeur a été programmé en DOM (Document Object Model) et Javascript. L'utilisateur peut cliquer sur un nœud pour modifier les informations de ce nœud, et manipuler la structure arborescente : ajouter un nœud, supprimer une branche, changer l'attachement d'une branche, etc.

Il y a une limitation importante : cet éditeur ne traite qu'un arbre UNL, c'est-à-dire un graphe UNL acyclique et sans scope.

L'utilisateur peut manipuler la représentation graphique ou textuelle, le changement sera reporté de l'un sur l'autre. Le résultat d'édition peut être sauvegardé sous forme XML.

Par rapport aux éditeurs précédents, cet éditeur de base est le premier qui est basé sur XML.

La structure de cet éditeur est la suivante :

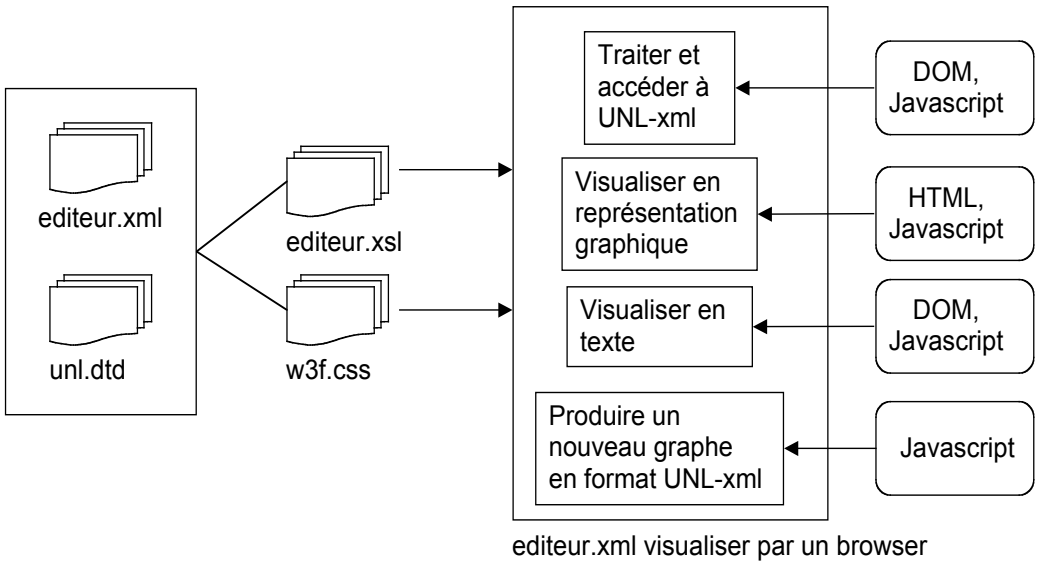


Fig. D-9 Structure de l'éditeur UNL de base

Voici deux images de cet éditeur :

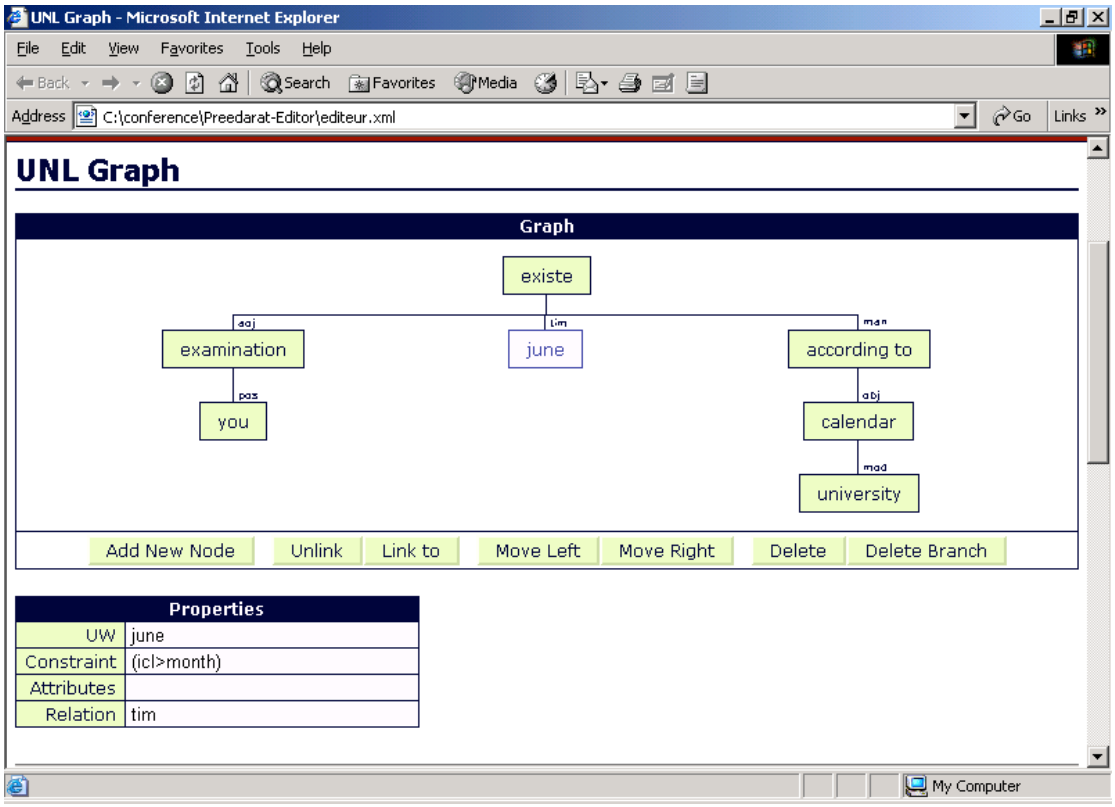


Fig. D-10 Information sur un nœud

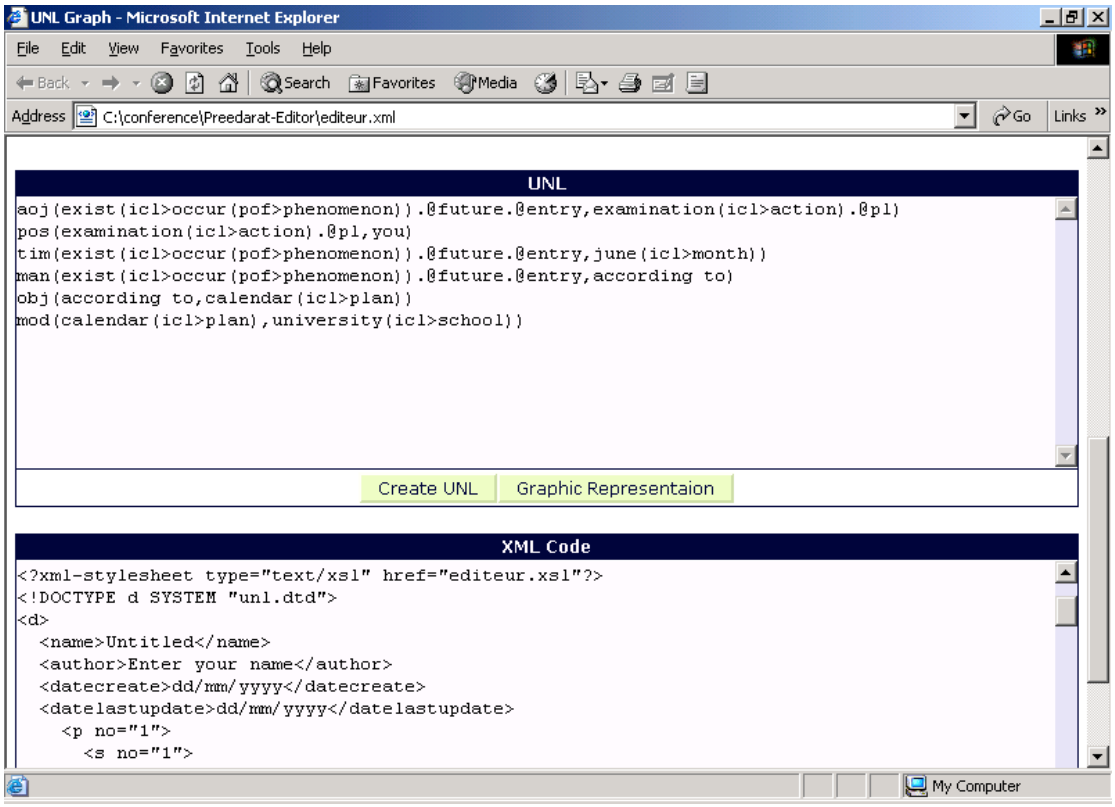


Fig. D-11 Génération du format UNL-xml

Durant l'été 2003, Stéphane Helme a réalisé un autre **éditeur UNL graphique** pendant son stage d'été de DESS. Cet éditeur est écrit en Java.

L'interface de cet éditeur se compose de trois parties : en haut, il y a des boutons pour l'édition de nœud, de relation, de restriction et de scope, ainsi que pour l'ouverture et la sauvegarde d'un fichier. Au milieu, on trouve le graphe UNL en représentation graphique, et en bas, le graphe UNL sous forme textuelle.

Cet éditeur inclut l'accès à un dictionnaire anglais-français et à la KB du centre UNL. Il peut donc proposer à l'utilisateur des UW en cours d'édition. L'utilisateur peut taper un mot anglais ou français pour trouver l'UW correspondante. En plus, cet éditeur permet à l'utilisateur de parcourir la KB si l'utilisateur a envie d'utiliser une UW dans la KB qui est proche du sens que cherche l'utilisateur. Cet éditeur peut aussi gérer les scopes, et tester la validité des relations et des restrictions.

Comme la taille de la KB s'accroît tous les jours, il faut télécharger la KB régulièrement.

Voici l'interface de cet éditeur :

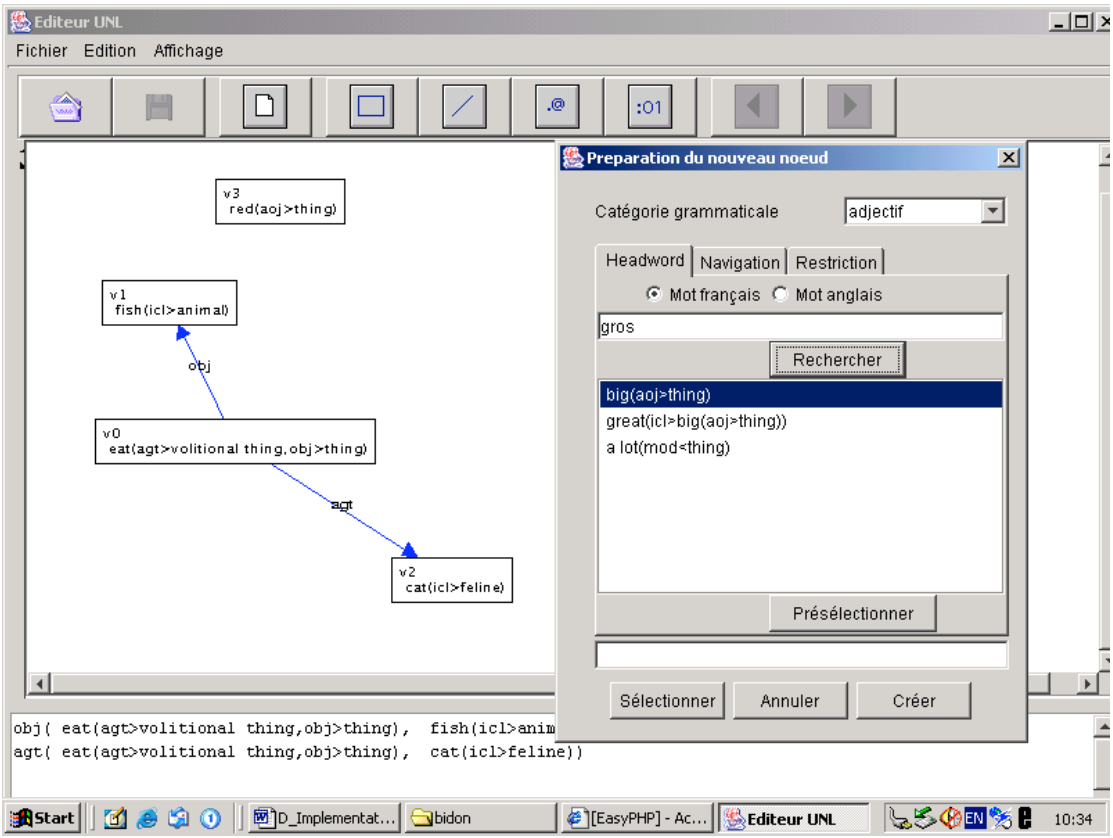


Fig. D-12 UW proposées par l'éditeur UNL graphique

Cet éditeur est écrit en Java, mais il utilise des bibliothèques de Jbuilder (l'environnement de développement Java de Borland) donc sa portabilité n'est pas totale. De plus, il n'est pas encore assez stable.

2.1.4 Déconvertisseur multilingue synchrone

Dans son stage précédent, Preedarat Jitkue a aussi réalisé un déconvertisseur UNL multilingue synchrone. Ce déconvertisseur permet à l'utilisateur d'envoyer un graphe UNL à plusieurs déconvertisseurs et de visualiser les résultats en parallèle.

La figure suivante montre la structure générale de ce déconvertisseur multilingue synchrone. L'interface est codée en Unicode et en trois langues : thaï, français, anglais. Il y a trois groupes de fichiers HTML en parallèle, un par langue. Il y a trois modules principaux :

un module de commande de déconversion et de détection des serveurs (status.pl),

un module de prétraitement des données (preload.pl),

un module de déconversion et de sortie des résultats (decon.pl).

Chaque fois que ce programme est appelé, il détecte les états des serveurs de déconversion, et informe l'utilisateur de la disponibilité de ces déconvertisseurs.

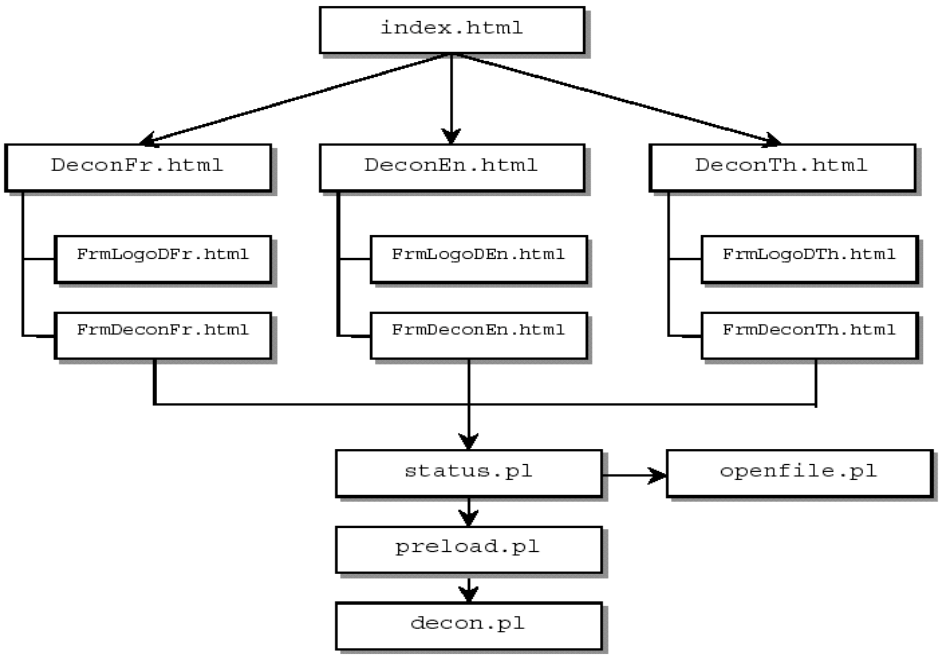


Fig. D-13 Structure du déconvertisseur multilingue synchrone

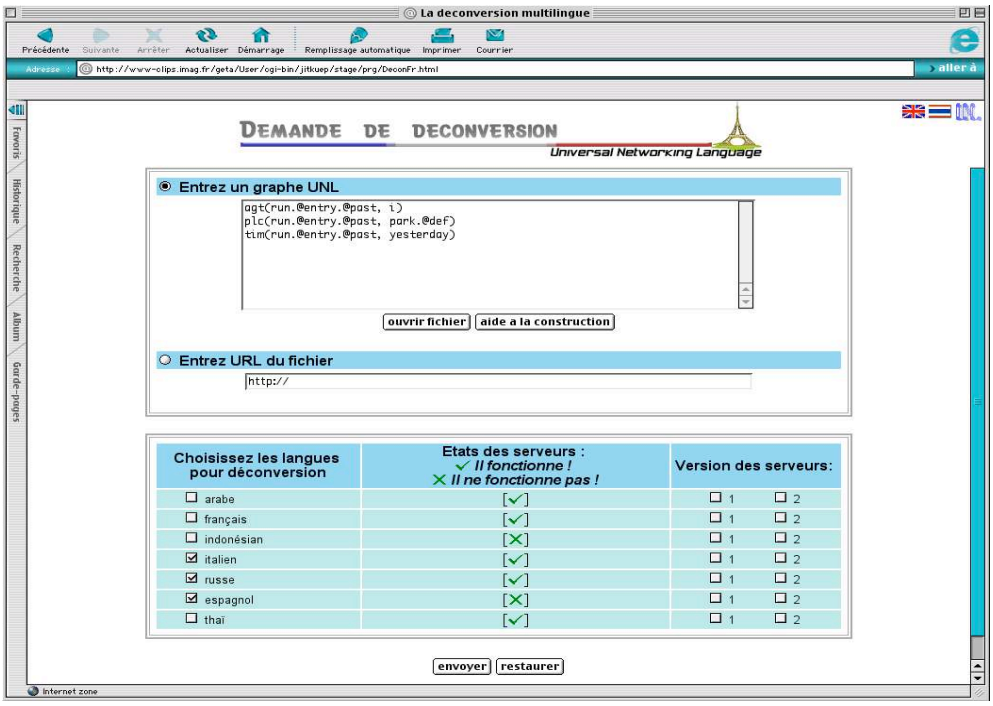


Fig. D-14 Déconvertisseur multilingue synchrone

L'utilisateur entre le graphe UNL et choisit les langues dans lesquelles il veut déconvertir. Il est possible qu'une langue ait plusieurs déconvertisseurs (par exemple, il existe deux déconvertisseurs anglais, celui de Moscou, officieux, et celui du centre UNL). L'utilisateur peut aussi choisir plusieurs déconvertisseurs pour une langue. Voici le résultat d'une déconversion multilingue synchrone.

Fig. D-15 Résultat de déconversion multilingue synchrone

Il faut changer le codage pour visualiser les réponses des serveurs russe et arabe.

Le point à améliorer dans ce programme est l'interaction avec les utilisateurs : il laisse l'utilisateur attendre sans donner l'état actuel de ce qui se passe. Donc, si un serveur se bloque, l'utilisateur doit attendre jusqu'à la limite de temps.

Il faudrait donner l'état actuel de la procédure à l'utilisateur, mais c'est difficile car rien n'est prévu pour ça dans les serveurs de déconversion. La seule chose que nous avons pu faire, c'est envoyer un tout petit graphe de test, avec une limite de temps correspondant au temps d'attente usuel pour ce graphe, pour chaque déconvertisseur. Si la déconversion n'est pas obtenue avant la limite, le serveur de déconversion correspondant est réputé indisponible.

2.1.5 Consultation de dictionnaires UNL-LN

La consultation de dictionnaires UNL-LN se fait par deux moyens. Si le dictionnaire existe sur notre serveur, un programme Perl trouve les bons articles. Sinon, un CGI ajoute les balises nécessaires et envoie une requête aux déconvertisseurs, s'ils peuvent déconvertir un graphe réduit à une seule UW. Pour l'instant, il n'y a que les déconvertisseurs français, russe et italien qui le peuvent. Le résultat de la consultation est affiché dans une autre fenêtre.

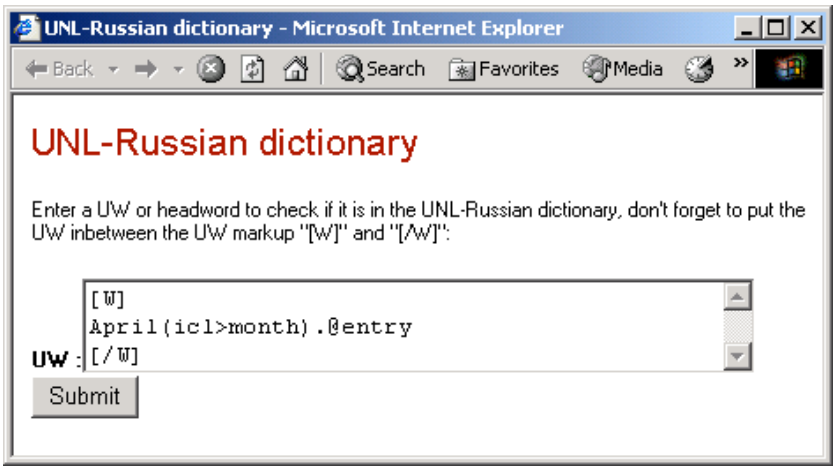


Fig. D-16 Consultation du dictionnaire UNL-russe

Le format de dictionnaire UNL-LN a été proposé par le centre UNL, et donc il n'est pas difficile de consulter les dictionnaires locaux. La difficulté est de produire la bonne forme de requête pour chaque déconvertisseur distant. En effet, bien que les balises pour spécifier une UW unique ([W] et [/W]) soient définies dans les spécifications du langage UNL, elles ne sont pas acceptées par tous les déconvertisseurs.

2.1.6 XML-isation de documents UNL

Nous avons déjà dit pourquoi il était souhaitable de « XML-iser » les documents UNL. Nous présentons ici le format UNL-xml (Unl-xml.1 pour la visualisation et UNL-xml.2 pour le traitement), puis le passage de UNL-html.1 à UNL-xml, et enfin la technique de visualisation utilisée.

2.1.6.1 Document UNL-xml

Dans la conception du centre UNL, un document UNL-html n'est qu'un intermédiaire pour stocker les données ; les utilisateurs visés ne sont que des utilisateurs qui veulent voir le document dans une certaine langue. Mais un document UNL peut servir à plus que ça, s'il est XML-isé. En effet :

- XML est assez standardisé et il y a beaucoup d'outils associés.
- Une fois qu'un document est XML-isé, il est facile de produire les formes de sortie souhaitées : forme textuelle « brute », format UNL-html, et d'autres formats (RTF par exemple), etc.
- Un document UNL-xml est facile à échanger, fusionner, et stocker dans une base de données.
- Il est facile de produire des variantes plus détaillées de UNL-xml selon nos besoins, par exemple, pour l'éditeur graphique. Il suffit d'ajouter des balises et de changer la DTD.
- Comme le format UNL-xml utilise Unicode (UTF-8), il est facile de visualiser tout ou partie des versions linguistiques qu'il contient.
- Il est facile de passer de UNL-html à UNL-xml.

La forme UNL-xml que nous proposons peut être exprimée dans la structure arborescente suivante, que nous appelons « forme UNL-xml de base » ou « forme **UNL-xml.1** ». Ici, les symboles utilisés par UNL et ayant des rôles spéciaux en XML, comme « < » et « > », sont laissés tels quels. La DTD de ce document est donnée dans l'Annexe B.

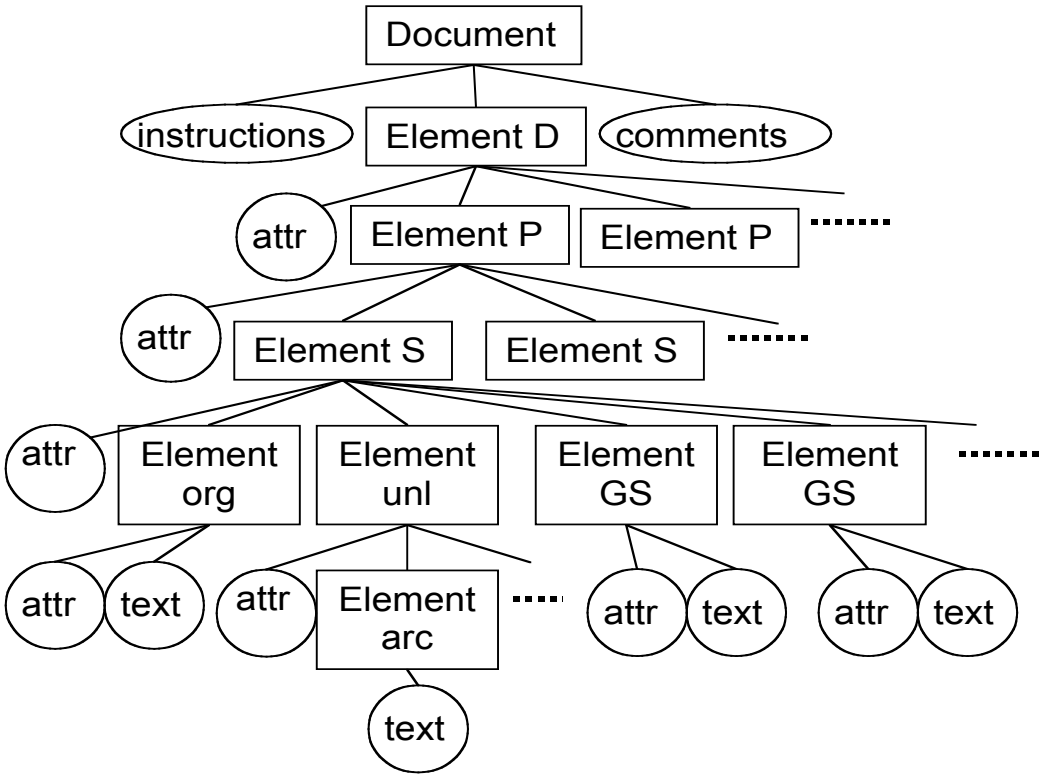


Fig. D-17 Structure d'un document UNL-xml.1

Pour les traitements par des outils XML, il faut encore échapper ces caractères particuliers (ou les remplacer par des entités). Nous appelons ce deuxième format « **UNL-xml.2** ».

Nous avons donc finalement quatre formats, comparés dans le tableau suivant.

.

	.1 (caractères spéciaux tels quels)	**.2 (caractères spéciaux traités)**
UNL-html	**stocké**	**traité**
	*proposé par le centre UNL. *format standard pour l'échange entre les équipes.	*compatible avec la syntaxe HTML. *affichable correctement sous un navigateur web.
UNL-xml	**visualisé**	**stocké et traité**
	*transfert direct depuis un document UNL-html.1. *ce que voit l'utilisateur sous un navigateur.	*compatible avec la syntaxe XML. *pour les autres applications liées au format XML.

Tableau D-2 Formats de document UNL

Voici un document UNL-xml.2 visualisé par l'éditeur textuel Notepad :

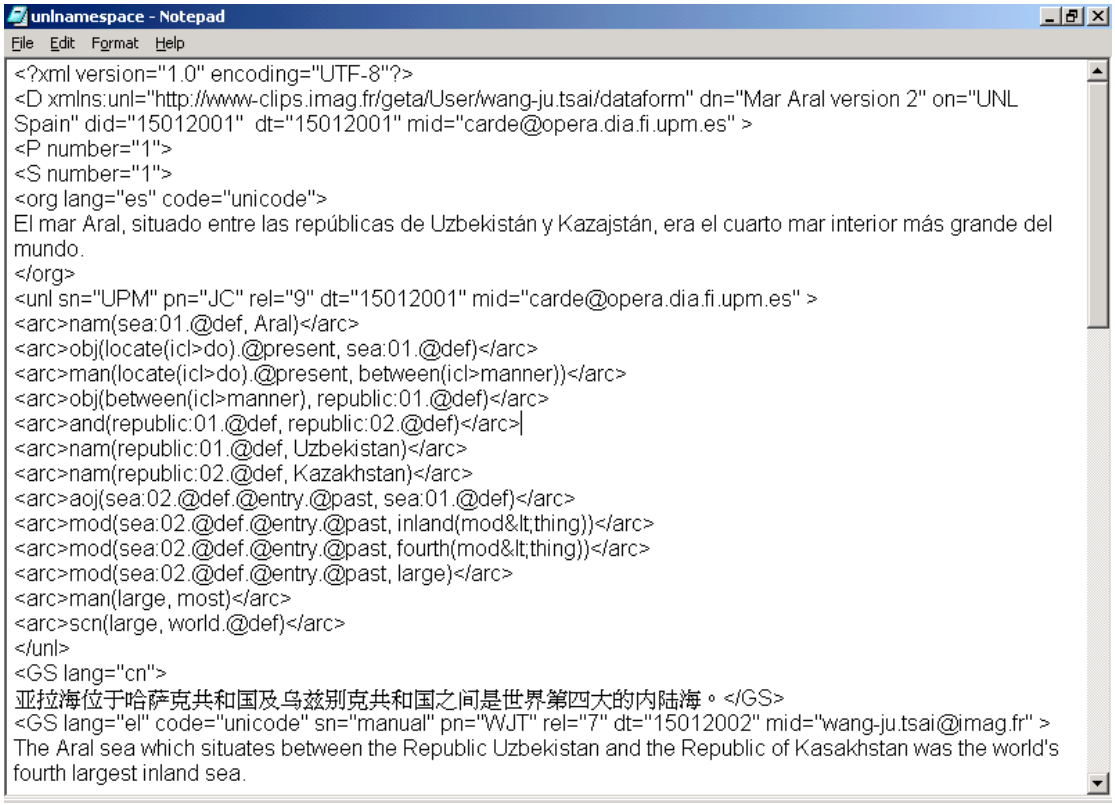


Fig. D-18 document UNL-xml.2 visualisé tel quel

Voici le même document UNL-xml.2 visualisé par le navigateur IE6.0 :

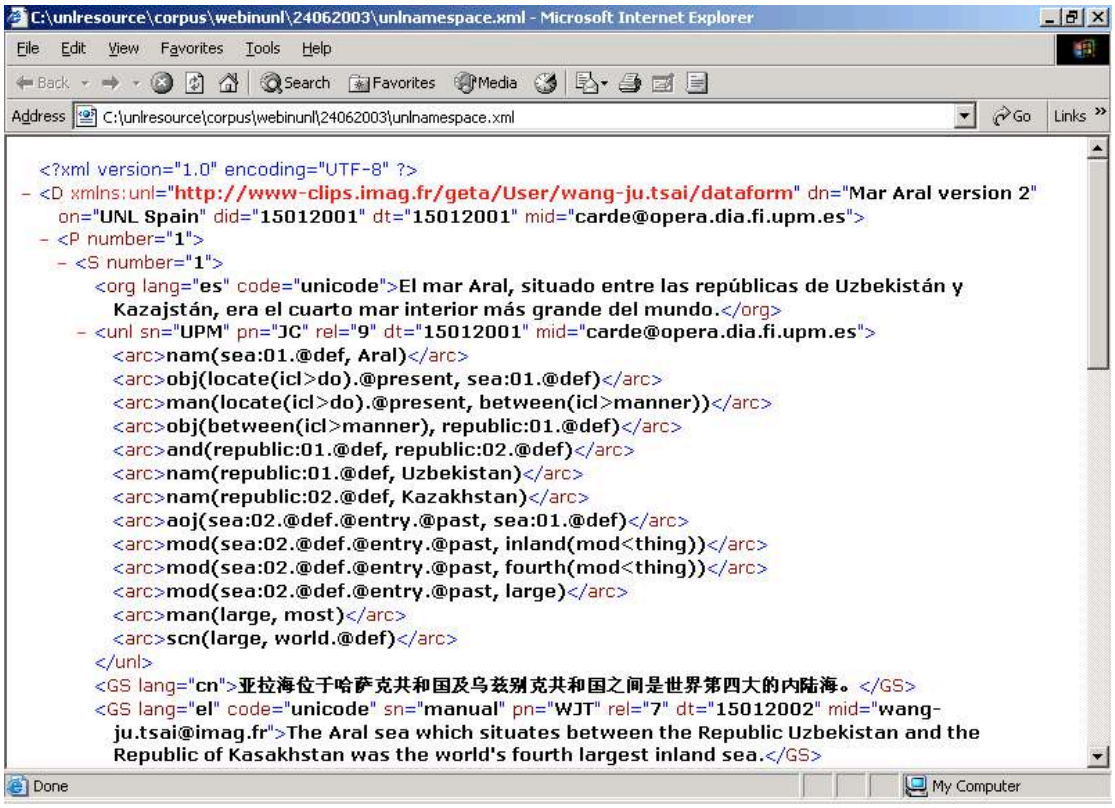


Fig. D-19 un document UNL-xml.2 visualisé par IE.6

Voir l'Annexe C pour un exemple complet du fichier UNL-xml.2.

Le principe du passage de UNL.html.1 à UNL-xml.2 est assez simple :

remplacer les caractères « < » et « & » dans les données textuelles ou les valeurs d'attribut par < et &amp. Ces deux caractères sont interdits dans les données textuelles XML,

changer les balises [] et {} en <>,

changer les attributs suivis par deux points : [P :1] en <P nombre="1">,

changer les attributs de codage : {org:unicode} en <org code ="unicode">,

transformer les étiquettes de langue en attributs de l'élément GS (generated sentence),

transformer les attributs qui restent : {es :sn=UPM} en <sn=="UPM">,

ajouter l'élément racine « document » et ajouter les commentaires et l'adresse de référence de l'espace de noms.

ajouter l'espace de noms « unl : » devant tous les éléments.

enlever les balises spécifiques à HTML comme <HTML>, </HTML> ; <HEAD>, </HEAD>, <BODY>, </BODY>.

A partir de cette forme, nous pouvons ajouter d'autres renseignements ou commentaires, simplement en changeant la DTD et en ajoutant des balises selon le besoin.

Dans notre éditeur de graphes UNL de base, par exemple, le graphe UNL est en fait détaillé jusqu'au niveau de chaque composant d'UW pour faciliter l'affichage de DOM par un navigateur. Ce genre de découpage en détail est nécessaire pour la coédition, si nous voulons garder dans le document les correspondances entre textes et graphes (et éventuellement les treillis AMS et les arbres UNL), et éventuellement toutes les correspondances possibles. De plus, nous devons ajouter des balises pour chaque utilisateur individuel, pour garder ses préférences et ses différentes versions.

Voici une image (du même graphe que celui de la Fig. D-19) de notre forme d'UNL-xml.2 pour la maquette de coédition sous IE 6.0 :

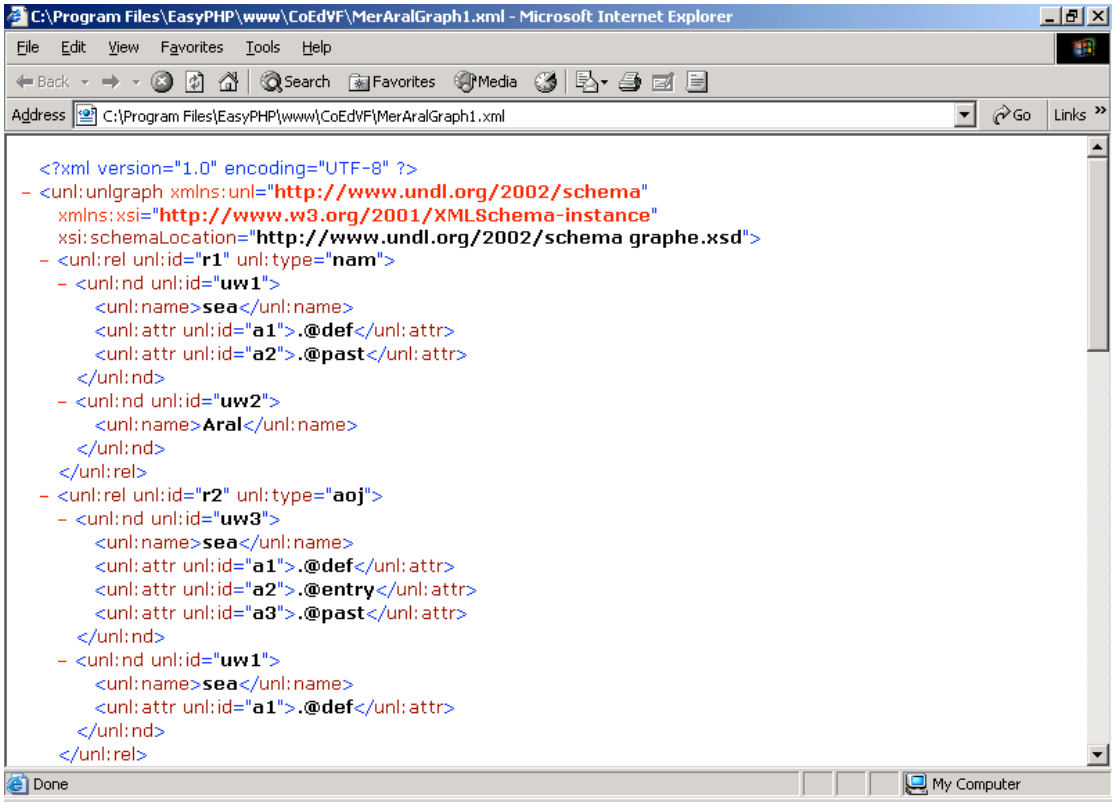


Fig. D-20 document UNL-xml.2 balisé plus en détail pour la maquette de coédition

2.1.6.2 Visualisation d'un document UNL-xml

Nous avons présenté le moyen proposé par le centre UNL pour visualiser un document UNL-html dans la section B.2.4.2.1. Une autre manière de visualiser un document UNL est de partir du format UNL-xml et d'utiliser des XSLT (eXtensible Stylesheet Language Transformation).

A partir de la forme UNL-xml, il est en effet beaucoup plus facile de produire des présentations variées, au moyen de transformations ou « feuilles de style » écrites en XSLT, qu'à partir du format UNL-html.

Sous un navigateur web, l'utilisateur, en cliquant sur le bouton de la langue qu'il veut visualiser, passe un paramètre au navigateur et le navigateur lie dynamiquement le document UNL-xml et la XSLT pour l'affichage. Une démonstration de cette visualisation se trouve sur le site SWIIVRE-UNL [SWIIVRE-UNL].

Voici la structure de notre visualiseur UNL-xml :

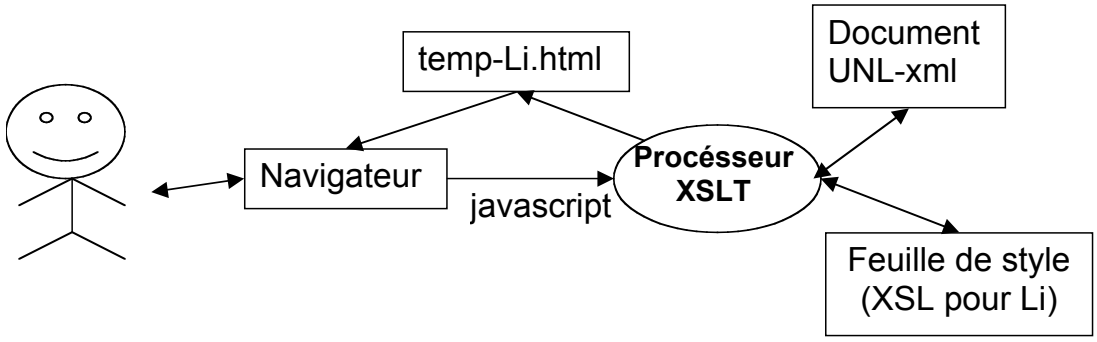


Fig. D-21 Structure du visualiseur UNL-xml

Voici quelques images de ce visualiseur UNL-xml.

En haut de la partie gauche de l'écran, l'utilisateur voit une liste de documents UNL-xml, et en bas la liste des langues. Le système propose en ce moment 14 langues (anglais, français, espagnol, italien, allemand, chinois, indonésien, thaï, arabe, hindi, russe, japonais, portugais, et letton), qui sont les langues que nous pouvons trouver dans les corpus UNL. L'utilisateur clique simplement sur le document et la langue, et le résultat est visualisé dans la partie droite de la fenêtre.

Dans l'image suivante, l'utilisateur a choisi de visualiser le document « Org-Information » en thaï.

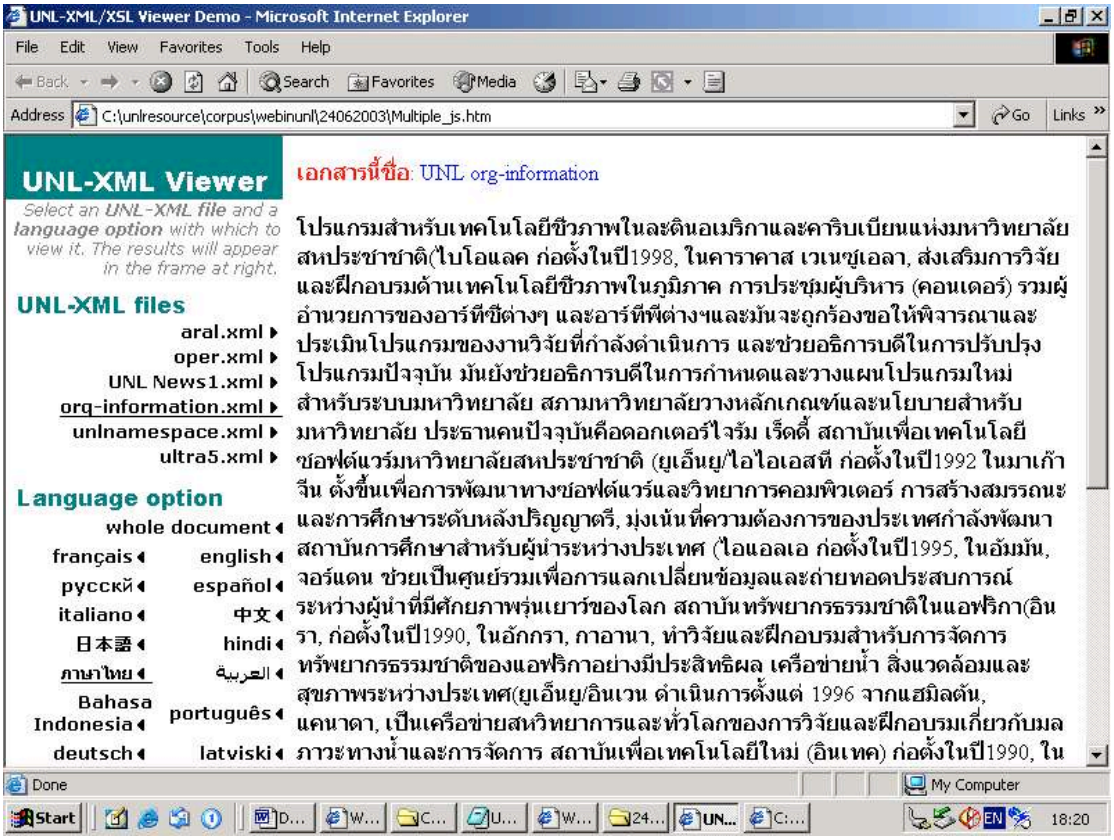


Fig. D-22 Visualisation d'un document UNL-xml en thaï

Puis l'utilisateur visualise ce document en arabe :

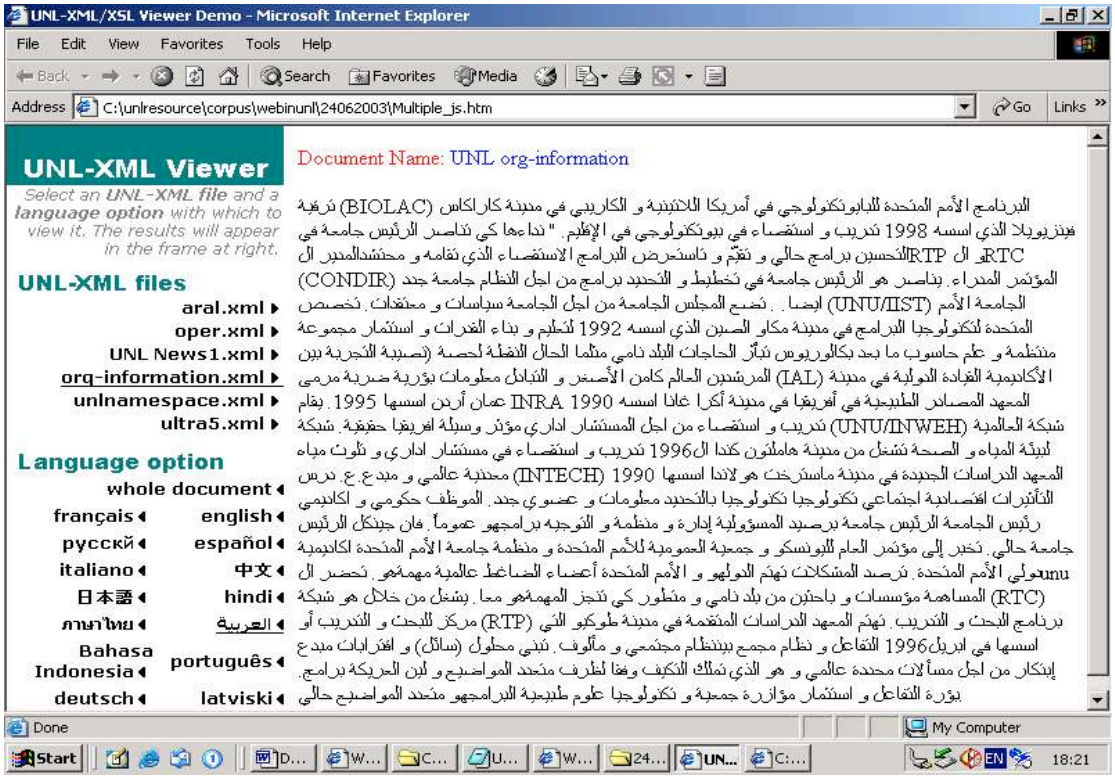


Fig. D-23 Visualisation d'un document UNL-xml en arabe

Enfin, l'utilisateur peut visualiser le document UNL-xml entier si ça l'intéresse :

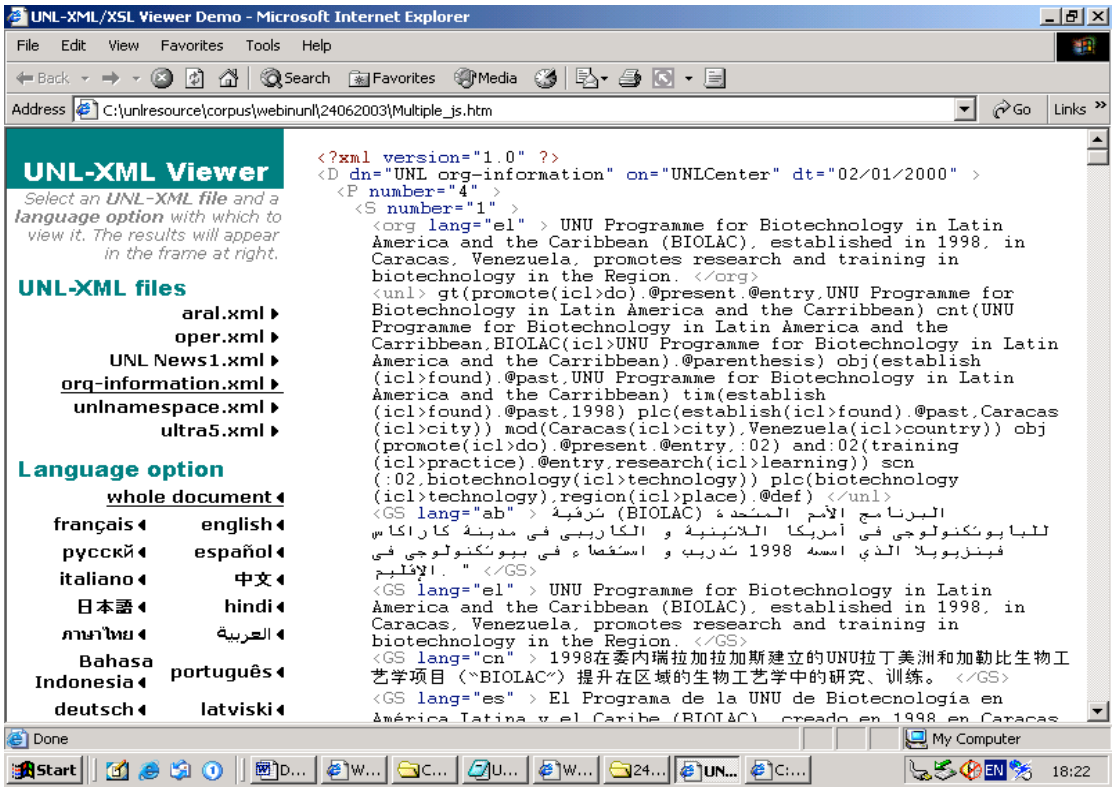


Fig. D-24 Visualisation d'un document UNL-xml entier

Voici un extrait du code Javascript de cette procédure. Les deux fonctions source.load et style.load chargent le fichier XML et la XSLT appropriée. Le changement de fichier XML ou XSLT se réalise par la fonction changeXML et changeXSL au moment où l'utilisateur clique sur les boutons de document et de langue. Voici le code de ces deux fonctions et la partie du code HTML qui les appelle :

```
//-- Script pour changer le fichier XML et XSLT.
//--
//-- Puis qu'on ne changer chaque fois qu'un seul fichier
//-- (soit fichier XML ou XSLT), on utilise le même booleén
//-- « viewingSrc » dans ces deux fonctions.
//--
//-- Les deux paramètres « changeXML » et « changeXSL » s'en
//-- servent comme le tampon pour sauvegarder la valeur
//-- précédente de l'un et de l'autre.

function changeXML(xmldoc)
    {
      if (viewingSrc)
      {
        styleURL = sourceURL;
      }
      sourceURL = xmldoc;
      source.load(sourceURL);
      if (viewingSrc)
      {
        viewingSrc = false;
        style.load(styleURL);
      }

      update();
    }
    function changeXSL(xsldoc)
    {
      if (!viewingSrc)
      {
        styleURL = xsldoc;
        style.load(styleURL);
      }
      else
      {
        sourceURL = xsldoc;
        source.load(sourceURL);
      }

      update();
    }

//-- quand l'utilisateur clique sur ce bouton
//-- le document org-information est choisi

<DIV CLASS="button"
           onMouseOver="over(this)"
           onMouseOut="out(this)"
```

```
            onClick='changeXML("org-information.xml");
            select("xml",this)'>
          org-information.xml<SPAN CLASS="arrow">4</SPAN>
        </DIV>

//-- quand l'utilisateur clique sur ce bouton
//-- la langue arabe ainsi la XSLT show2ab.xsl est choisie

//load xsl ab
<DIV CLASS="button"
            onMouseOver="over(this)"
            onMouseOut="out(this)"
            onClick='changeXSL("show2ab.xsl");
            select("xsl",this)'>
          _______<SPAN CLASS="arrow">3</SPAN>
    </DIV>
```

Le code complet de ce programme se trouve sur le site SWIIVRE-UNL et plus de documentation et d'outils associés à XML sous l'environnement Windows™ se trouvent sur [MSDN].

Voici une XSLT pour extraire la version russe :

```
<?xml version='1.0'?>
<!-- pour visualiser sous HTML , 03/04/2002, WJT  -->
<xsl:stylesheet xmlns:xsl="http://www.w3c.org/TR/WD-xsl">
<xsl:template match="/">
<!-- ce template s'applique sur le document xml entier  -->
  <html>
  <body>
  <th><font color="red">___ _________: </font></th>
  <font color="blue"><xsl:value-of select="/D/@dn"/></font>
  <br></br>
  <xsl:for-each select="//GS[@lang='ru']">
     <xsl:value-of select="."/>
  </xsl:for-each>
  </body>
</html>
</xsl:template>
```

Dans notre approche, il faut avoir une XSL pour chaque langue.

L'avantage de cette approche est que l'utilisateur n'a pas besoin de télécharger et d'installer une application. Il suffit d'avoir un navigateur qui peut lire Unicode et qui est compatible avec le DOM, ce qui est le cas des navigateurs récents. Une XSLT plus sophistiquée peut afficher le document comme on le veut, par exemple, avec une mise en page.

Un point à améliorer dans ce module est de permettre à l'utilisateur de choisir dynamiquement les documents UNL-xml et ses langues de préférence.

2.1.7 Documents UNL sur le web

On peut considérer les formes « document unique multilingue parfaitement aligné », comme UNL-html ou UNL-xml, comme des formes de travail et d'échange. Mais la

réalité est souvent plus complexe. Par exemple, un site web comme HEREIN [HEREIN] ou UNESCO [UNESCO] a déjà une organisation avec des documents monolingues parallèles, mais pas parfaitement alignés au niveau des phrases. On pourrait y associer un document non structuré UNL-xml, servant seulement à stocker les équivalences traductionnelles. N. Hajlaoui travaille sur cette idée dans le cadre de sa thèse [Hajlaoui 03].

Pour promouvoir le format UNL-xml, nous avons écrit un module sur SWIIVRE-UNL pour transformer un document UNL-html.1 en un document UNL-xml.2. L'utilisateur peut copier et coller un document UNL-html.1 entier dans le cadre. On peut demander le format avec ou sans espace de noms.

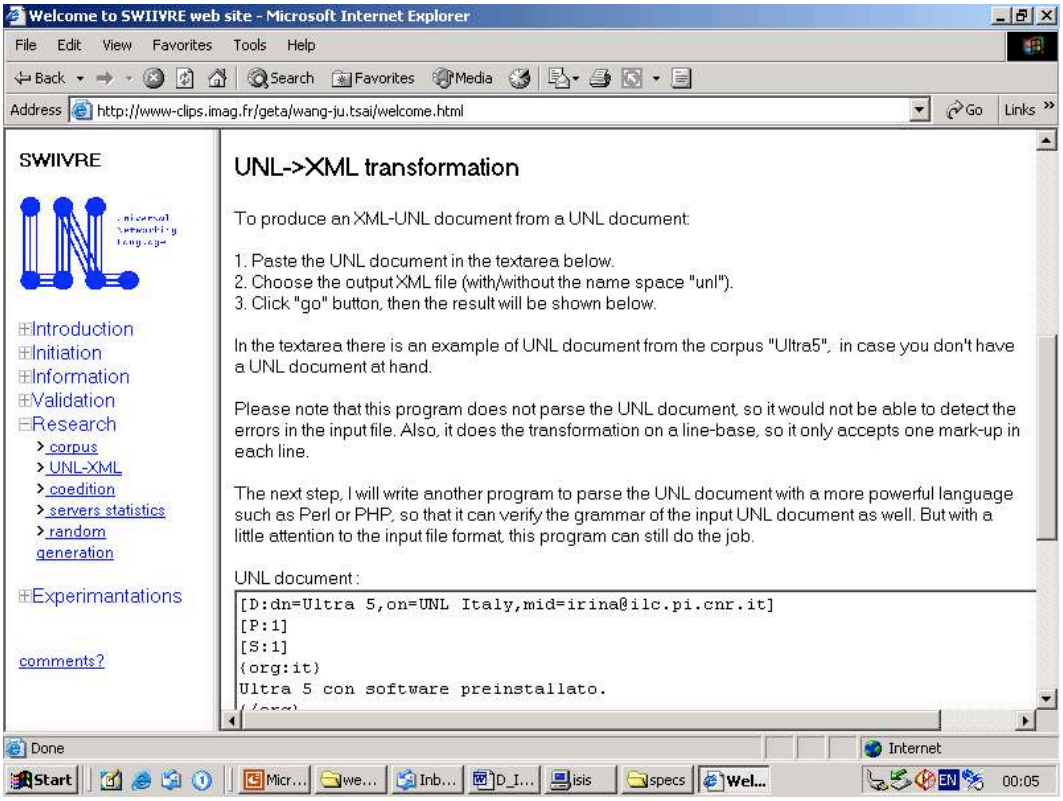


Fig. D-25 Transformation d'un document UNL-html.1 en UNL-xml.2

Voici le résultat :

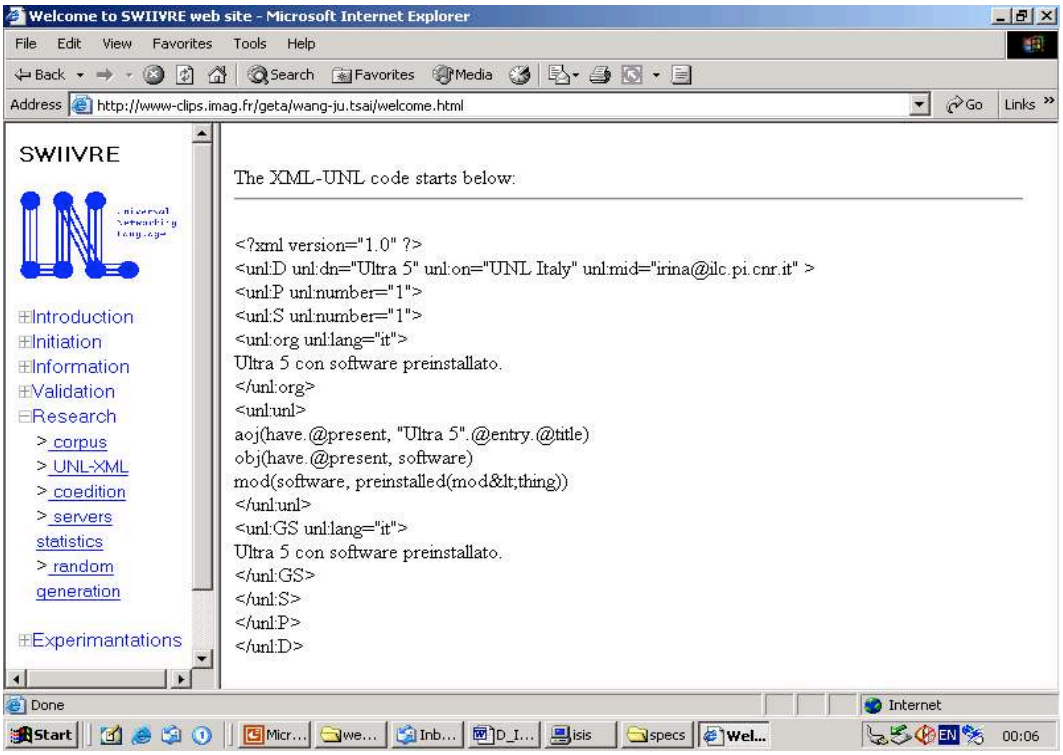


Fig. D-26 Résultat : document UNL-xml.2

Le module renvoie le résultat (un document UNL-xml.1) dans le même cadre.

Le programme est écrit en Perl, l'algorithme est le suivant :

a. lire le document UNL entier et les paires variable-valeur, puis les stocker,

b. ajouter l'en-tête HTML (puisque le document va être affiché dans une page web),

c. ajouter l'en-tête XML dans le document UNL,

d. lire ligne par ligne le document UNL et changer les balises en format XML,

e. échapper les caractères interdits par XML,

f. écrire le résultat dans le même cadre.

Un meilleur moyen de réaliser ce module serait de le développer en ANTLR [ANTLR] (« ANother Tool for Language Recognition », successeur de PCCTS « Purdue Compiler Construction Tool Set ») ou en YACC (Yet Another Compiler-Compiler) qui peuvent d'abord parser le document UNL et produire la forme spécifiée (notamment UNL-xml). Ce que fait notre programme consiste simplement à changer les balises dans un ordre heuristique.

Voici un extrait essentiel (la partie de changement des balises) de ce programme,

```
foreach $docline (@unldoc)
{    print "<BR>";
     $docline=~ s/\[(.*)\]/\<$1\>/g;
     $docline=~ s/\{(.*)\}/\<$1\>/g;
     $docline=~ s/\<(unl):/\<unl:$1 /g;
     $docline=~ s/\<unl\>/\<unl:unl\>/g;
```

```
        $docline=~ s/\<P:(\d+)/\<unl:P unl:number\="$1"/g;
        $docline=~ s/\<P\>/\<unl:P\>/g;
        $docline=~ s/\<S:(\d+)/\<unl:S unl:number\="$1"/g;
        $docline=~ s/\<D:/\<unl:D /g;
        $docline=~
s/\<\/(ab|cn|de|el|es|fr|id|hd|it|jp|pg|ru|th)\>/\<\/unl:GS\>/
g;
        $docline=~
s/\<(ab|cn|de|el|es|fr|id|hd|it|jp|pg|ru|th)\>/\<unl:GS
unl:lang="$1"\>/g;
        $docline=~
s/\<(ab|cn|de|el|es|fr|id|hd|it|jp|pg|ru|th)=(.*):/\<unl:GS
unl:lang="$1" unl:code="$2" /g;
        $docline=~ s/\<org:(..)=(.*)\>/\<unl:org unl:lang="$1"
unl:code="$2"\>/g;
        $docline=~ s/\<org:(..)\>/\<unl:org unl:lang="$1"\>/g;
        $docline=~ s/\<org\>/\<unl:org\>/g;
        $docline=~ s/(\<.*)(dn=)(.*?)[,|\>]/$1unl:$2"$3" /g;
        $docline=~ s/(\<.*)(on=)(.*?)[,|\>]/$1unl:$2"$3" /g;
        $docline=~ s/(\<.*)(did=)(.*?)[,|\>]/$1unl:$2"$3" /g;
        $docline=~ s/(\<.*)(dt=)(.*?)[,|\>]/$1unl:$2"$3" /g;
        $docline=~ s/(\<.*)(mid=)(.*?)[,|\>]/$1unl:$2"$3" /g;
        $docline=~ s/(\<.*)(sn=)(.*?)[,|\>]/$1unl:$2"$3" /g;
        $docline=~ s/(\<.*)(pn=)(.*?)[,|\>]/$1unl:$2"$3" /g;
        $docline=~ s/(\<.*)(rel=)(.*?)[,|\>]/$1unl:$2"$3" /g;
        $docline=~ s/\<(unl:GS|unl:unl|unl:D)(.*)/\<$1$2\>/g;
        $docline=~
s/\<(unl:GS|unl:unl|unl:D)(.*)\>(.*)\>/\<$1$2\>/g;
        $docline=~ s/\<\/(D|P|S|GS|unl|org)\>/\<\/unl:$1\>/g;
        $docline=~ s/\</&lt;/g;
        $docline=~ s/\>/&gt;/g;
        $docline=~ s/\"/&quot;/g;
        $docline=~ s/mod\&lt;thing/mod\&amp;lt;thing/g;
        print $docline;
    }
```

Pour que l'utilisateur ait une idée claire de la transformation UNL-html en UNL-xml, ce programme, au lieu de sauvegarder le résultat de transformation dans un fichier, l'affiche dans une fenêtre. L'utilisateur doit copier et coller le résultat dans un autre fichier s'il veut le sauvegarder.

Ce programme n'en est qu'à sa première version, et peut encore être amélioré sur les points suivants :

ouvrir directement un fichier,

sauvegarder le document UNL-xml directement dans un fichier spécifié,

permettre à l'utilisateur plus de choix sur l'espace de noms, l'en-tête XML,

valider d'abord le document UNL-html,

accepter un document UNL-html contenant des encodages différents,

transformer dans les deux sens, c'est-à-dire permettre de produire aussi un document UNL-html depuis un document UNL-xml,

combiner avec les autres modules, par exemple, l'éditeur UNL, la maquette de coédition ou le valideur UNL.

2.2 Maquette de coédition

2.2.1 Évolution de la maquette

Au tout début, nous avons conçu une interface de cette maquette dans [Boitet 02b, 02c]. Nous avons dessiné cette interface avec HTML simplement pour montrer l'idée et les scénarios de la coédition.

L'interface principale de coédition se compose de trois parties : en haut, les boutons des manipulations de fichier (ouvrir, quitter, sauvegarder, etc.) et les champs où l'utilisateur peut saisir du texte, ou cliquer sur le texte et initier la coédition. Au milieu, il y a une fenêtre pour montrer l'arbre UNL. En bas, on voit les versions de déconversion avant et après la coédition, en différentes langues.

Voici une image de cette première interface :

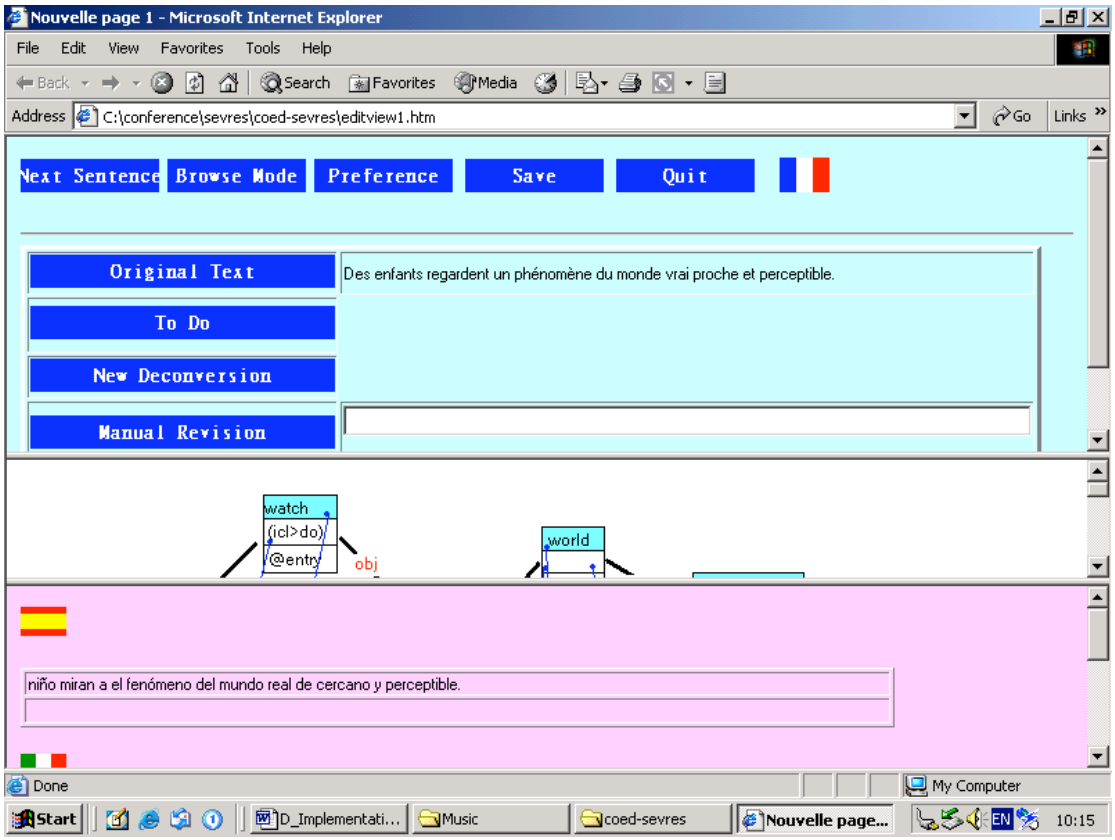


Fig. D-27 Première interface de coédition

La maquette de coédition que nous allons présenter maintenant a été réalisée pendant trois stages [Bernhard 02] [Helme 02,03] de deux mois chacun. Plus tard, nous avons apporté de petites modifications pour rendre la maquette plus proche de notre maquette idéale. Les résultats principaux sont :

l'implémentation de l'interface. Tous les éléments sont en place, y compris ceux permettant d'appeler ou de visualiser des fonctionnalités qui n'ont

pas encore été implémentées (par exemple□ boutons «□déconversion□», option de menu «□Sauvegarder□»). L'interface montre aussi le graphe UNL pour que l'utilisateur puisse mieux comprendre les rapports entre le texte et le graphe.

l'architecture client-serveur. Il y a une base de textes UNL-xml et des scripts PHP côté serveur pour gérer l'interaction entre les modules locaux (UNL-xml fichiers, applet) et distants (dictionnaire, AMS), et une applet Java côté client pour l'interface. Pour l'instant, cette maquette n'a pas encore été mise sur Internet, mais elle tourne sur le réseau local.

les traitements UNL-xml. Ils constituent une des partie les plus complexes du système, et fonctionnent. On peut extraire des informations dans les documents UNL-xml.

le scénario type a été réalisé.

la connexion à PILAF et au dictionnaire UNL-français et le traitement des résultats des requêtes correspondantes ont été réalisés.

Cependant, certaines fonctionnalités, qu'il est important de pouvoir montrer lors d'une démonstration du système, n'existent que de manière superficielle : on est par exemple obligé de recourir à des fichiers spéciaux (avec balises UNL-xml-coéd plus détaillé que UNL-xml), ce qui serait exclu dans la perspective d'une utilisation normale de l'applet.

Nous appelons cette maquette la version _ de la maquette de coédition.

En ce moment, nous travaillons sur une nouvelle maquette dans laquelle nous implémentons exactement les structures de données et l'algorithme que nous présentons dans la partie C. Cette nouvelle maquette est complètement écrite en Java. Quand elle sera finie, elle pourra montrer les scénarios types de la procédure de coédition, contacter les différents déconvertisseurs, et calculer vraiment la meilleur correspondance.

2.2.2 Introduction à la version _

Voici quelques images de la version _ de la maquette de coédition durant une session de coédition.

On lance d'abord EasyPHP pour créer l'environnement web local. Supposons que l'utilisateur est venu sur notre serveur et qu'il veut visualiser des documents UNL-xml.

Le système lui propose une liste de documents UNL-xml, et les versions disponibles de ces documents. Il y a ici trois documents UNL-xml, chacun en trois langues, parmi lesquelles l'utilisateur peut choisir.

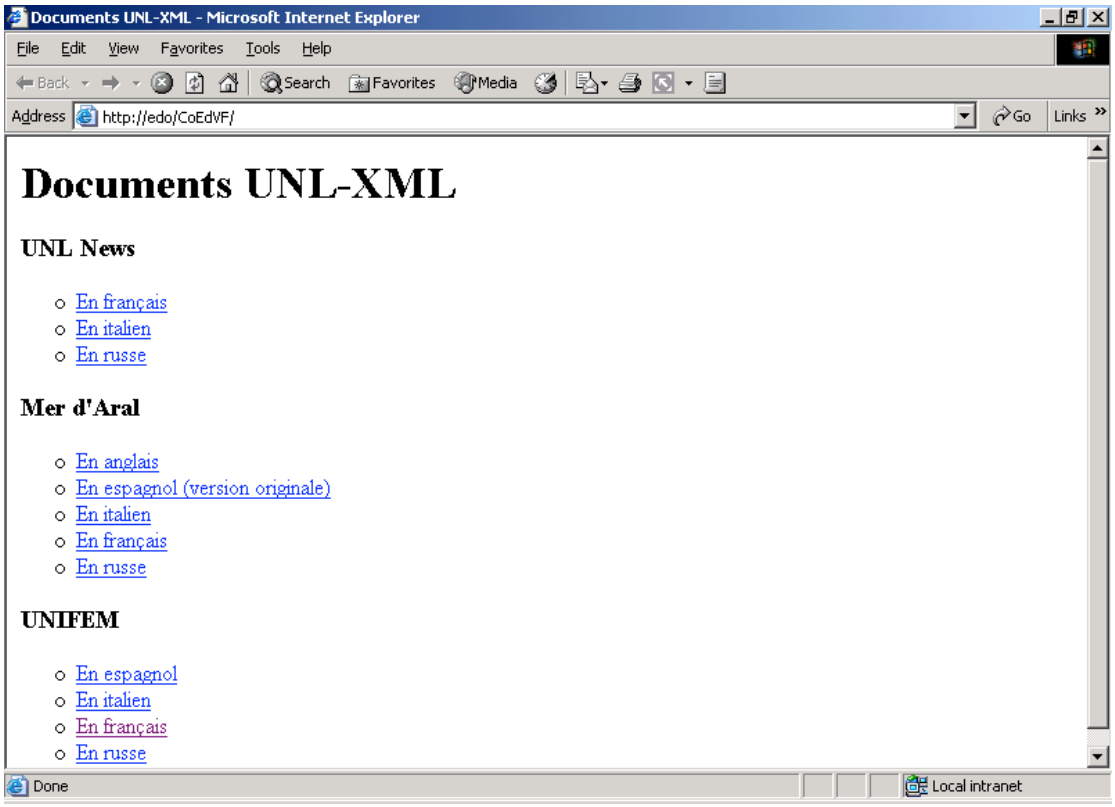


Fig. D-28 Documents UNL-xml à choisir

L'utilisateur choisit le document UNIFEM et la version française, et entre dans le mode de lecture. Le script PHP extrait et affiche le texte dans une fenêtre du navigateur.

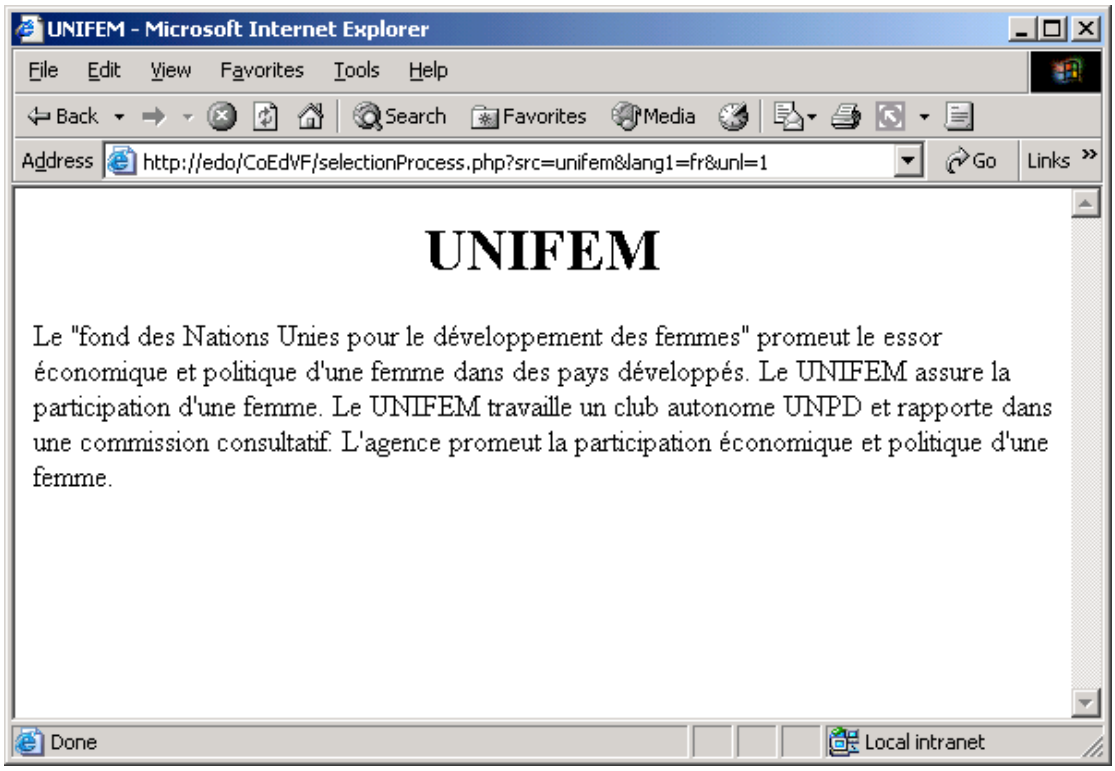


Fig. D-29 Lecture en français d'un document UNL-xml multilingue

Si l'utilisateur sélectionne un fragment quelconque du texte, la sélection est automatiquement étendue à la plus petite liste de phrases qui contient ce fragment. Cela est rendu possible par la forme du fichier html généré à partir du document UNL-xml : chaque phrase y est contenue dans un élément « span » contenant l'appel à une fonction Javascript adéquate, qui étend la sélection comme dit plus haut, et demande confirmation à l'utilisateur de son désir de la coédition.

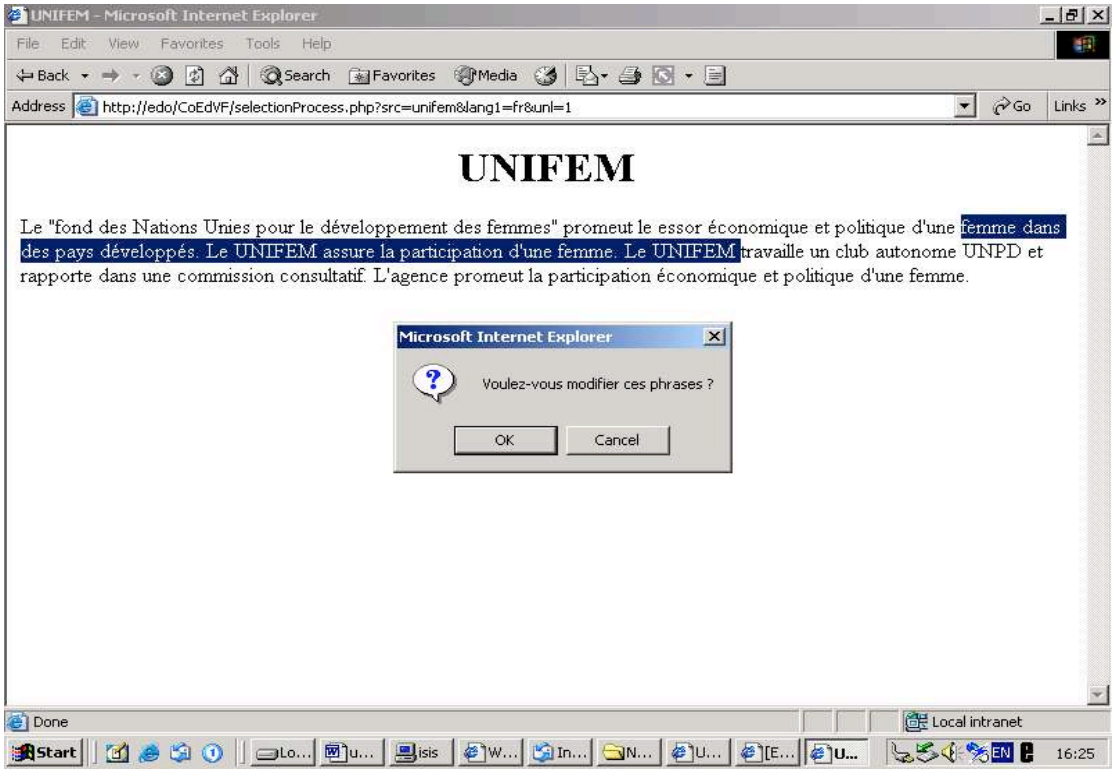


Fig. D-30 Sélection d'un fragment à coéditer

Une fenêtre d'une applet Java de coédition apparaît après la confirmation de l'utilisateur. L'utilisateur a ici choisi trois phrases et donc la procédure de coédition va s'appliquer une par une à ces trois phrases. L'interface permet à l'utilisateur de choisir la langue de travail (pour l'instant français et anglais) et de visualiser dans les autres langues que celle que l'utilisateur a choisie.

Il y a deux onglets principaux : texte et traitement. L'onglet texte affiche le(s) texte(s) et l'onglet traitement permet à l'utilisateur de coéditer le texte phrase par phrase.

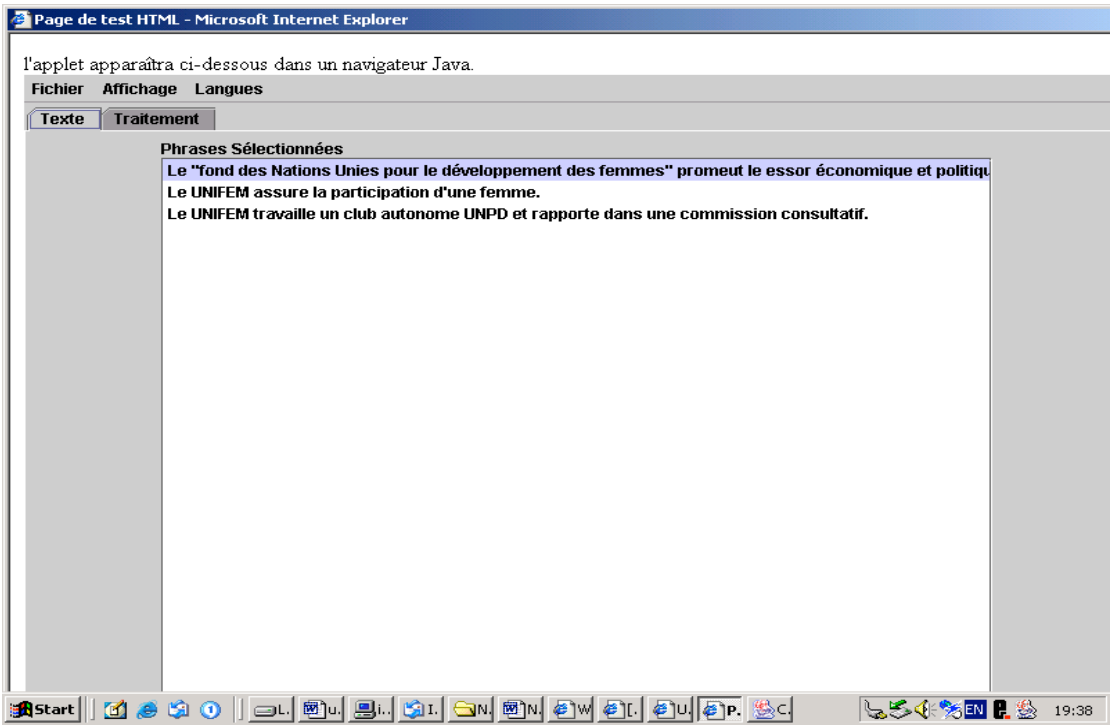


Fig. D-31 État initial de la coédition de trois phrases

Le mode de coédition comprend trois cadres : en haut le cadre du traitement de texte, au milieu le cadre de la représentation graphique, et en bas le cadre pour les autres langues.

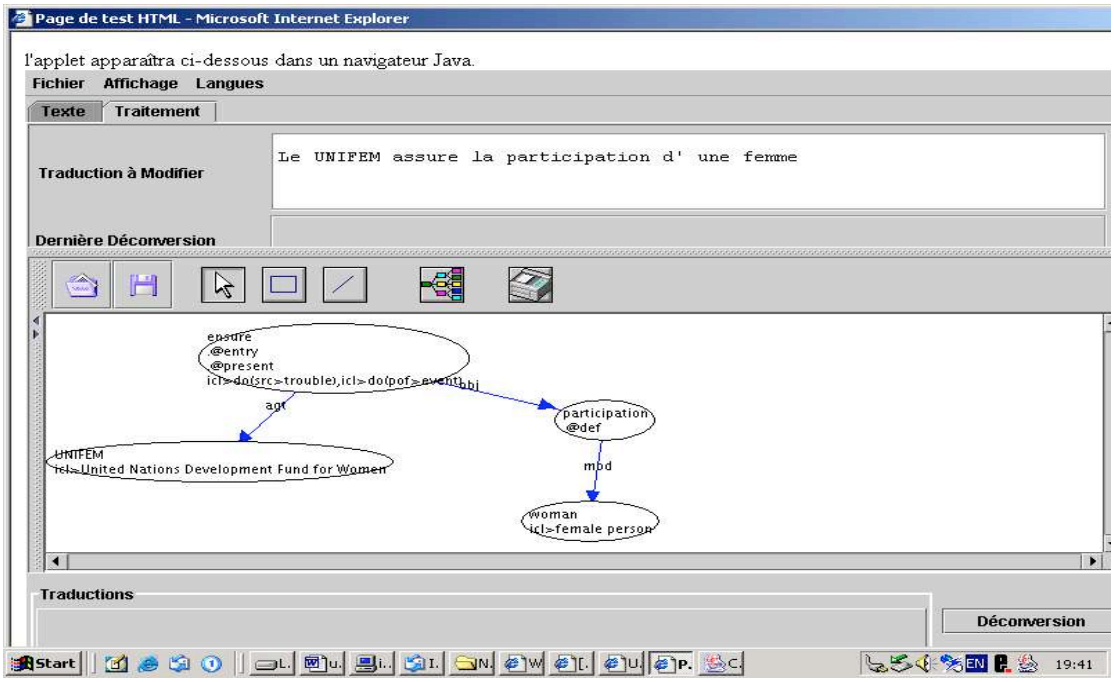


Fig. D-32 Trois cadres dans l'environnement de coédition

L'utilisateur peut choisir les autres langues, à visualiser en cliquant sur le bouton « langues ».

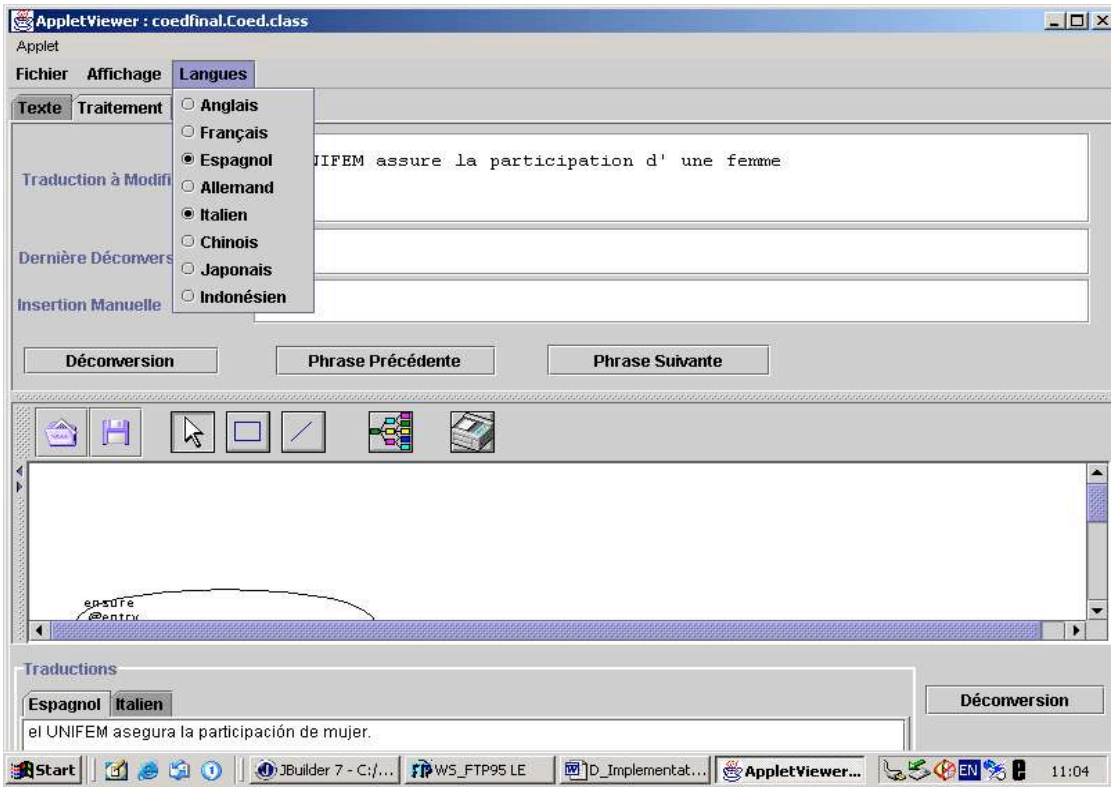


Fig. D-33 Choix de visualisation des autres langues

Ici, l'utilisateur a choisi l'espagnol et l'italien.

Il y a aussi un champ pour l'insertion manuelle si l'utilisateur juge qu'il est plus simple de taper entièrement une nouvelle phrase.

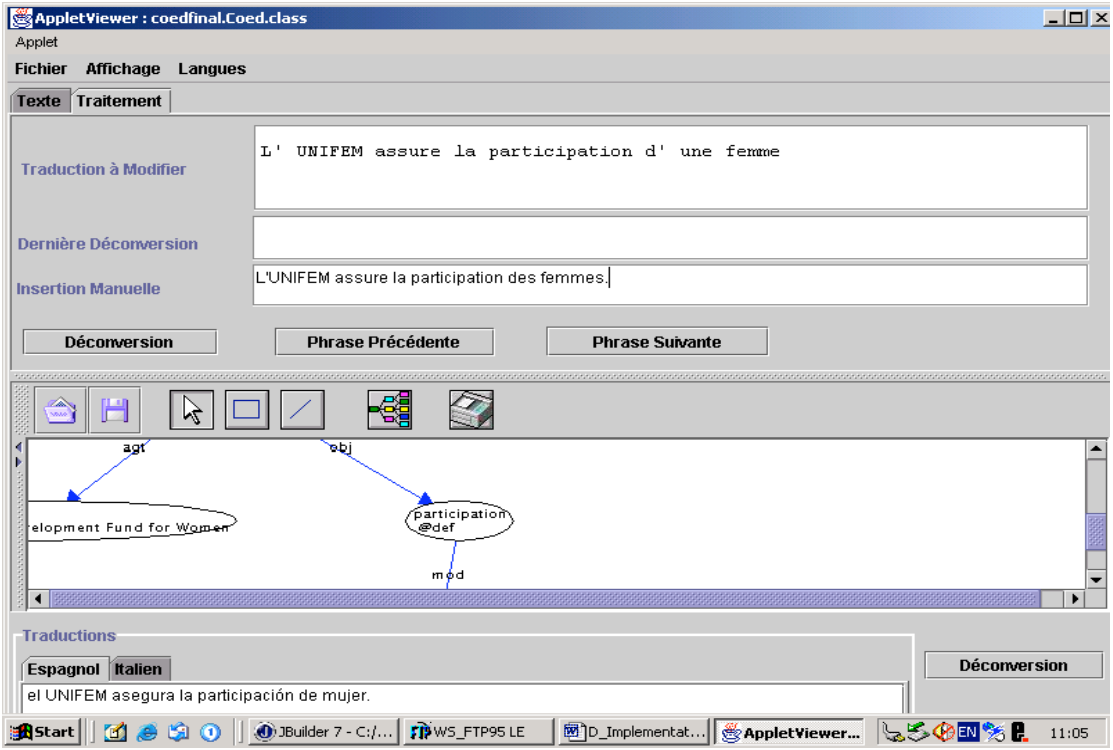


Fig. D-34 Insertion manuelle

L'utilisateur clique sur le mot qu'il souhaite modifier et le système lui propose en retour les modifications possibles.

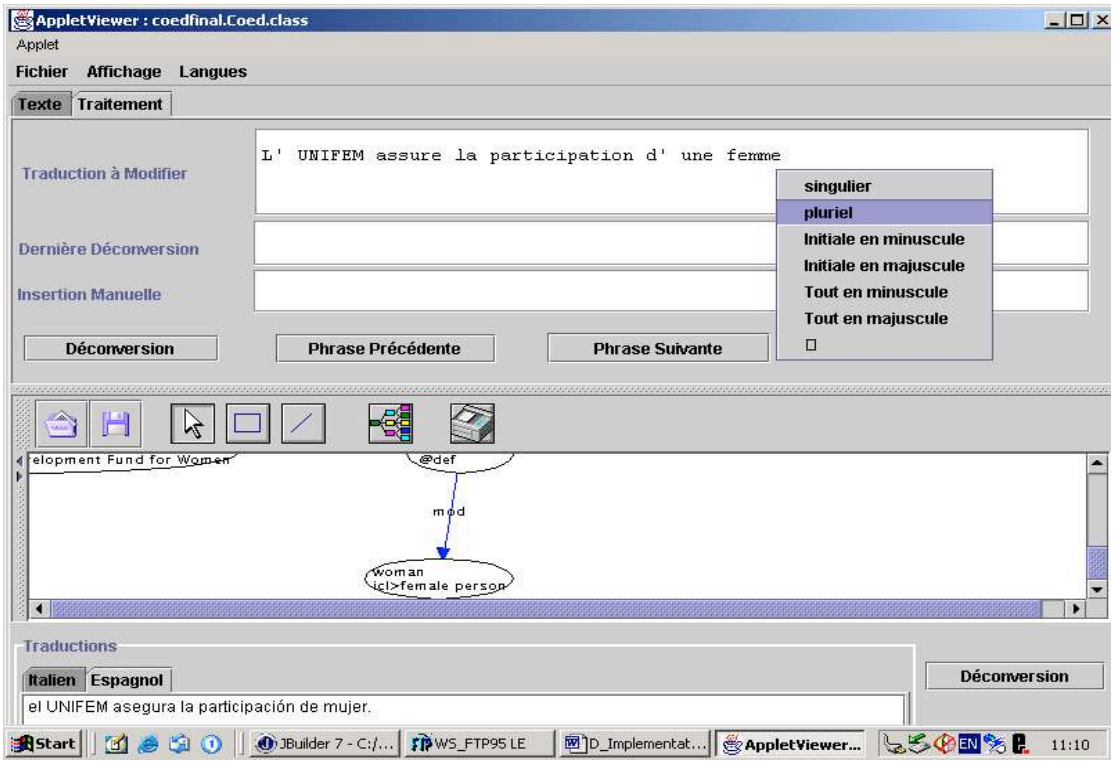


Fig. D-35 Modifications possibles proposées par le système

Le système montre le choix fait par l'utilisateur, et marque en même temps la modification correspondante sur le graphe.

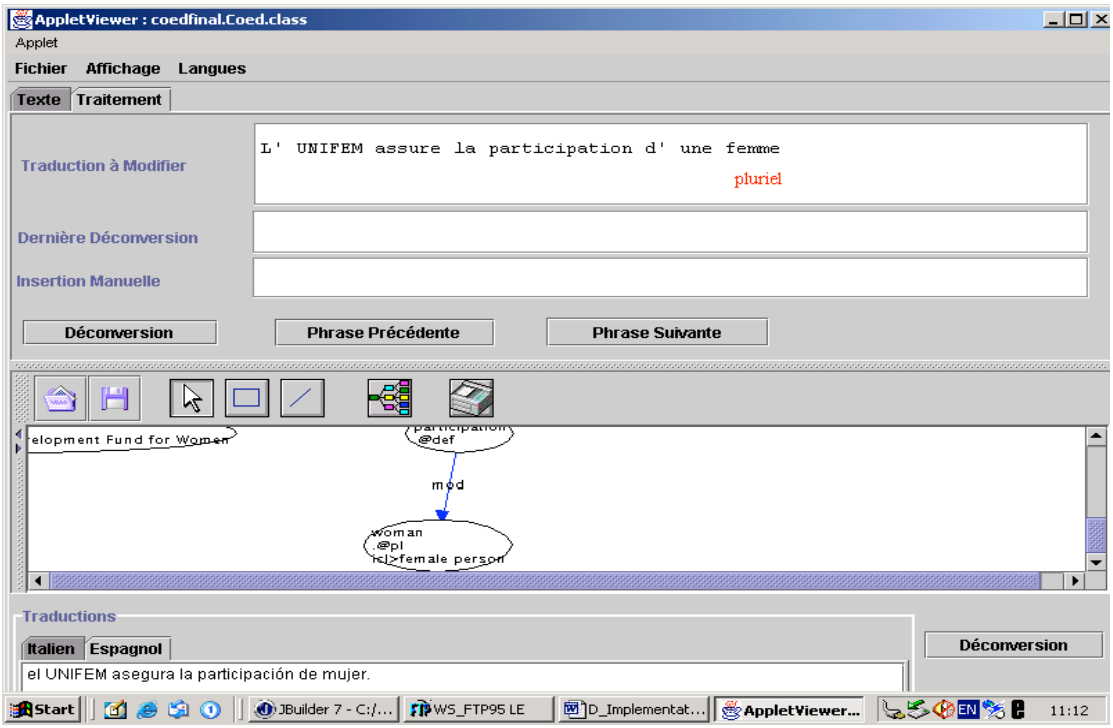


Fig. D-36 Modification faite

Une fois satisfait, l'utilisateur peut cliquer sur le bouton « Déconversion » et envoyer le nouveau graphe au déconvertisseur. Dans cette maquette, la communication entre l'applet et le déconvertisseur a été construite. Le résultat de la déconversion apparaît dans le champ « dernière déconversion ».

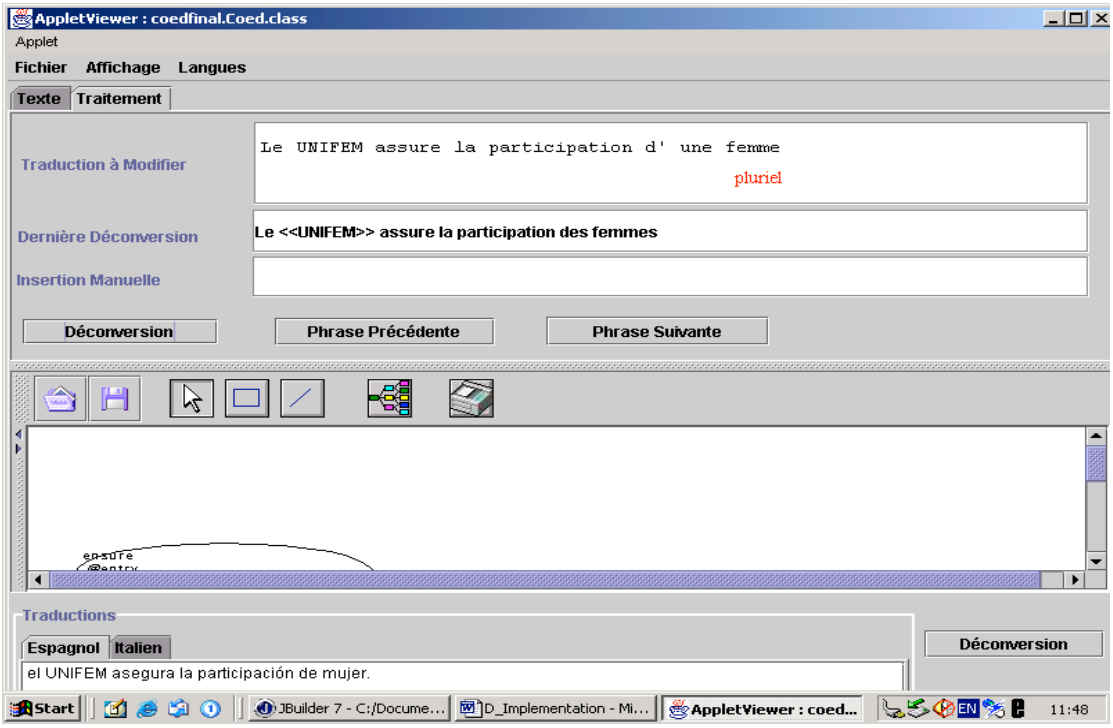


Fig. D-37 Récupération de la nouvelle déconversion

Le bouton « déconversion » en bas sert à déconvertir le graphe UNL modifié vers les autres langues, pour que l'utilisateur puisse comparer les résultats dans les autres langues.

Si l'utilisateur veut encore faire des modifications, il peut répéter la procédure ci-dessus. Par exemple, il clique sur le mot « assure » et le système lui propose les modifications possibles d'un verbe[21].

[21] Ici, les modifications proposées pour un verbe ne sont que tentatives. Le menu de proposition n'est pas facile à concevoir. Nous en discuterons plus tard en section D.2.2.5.

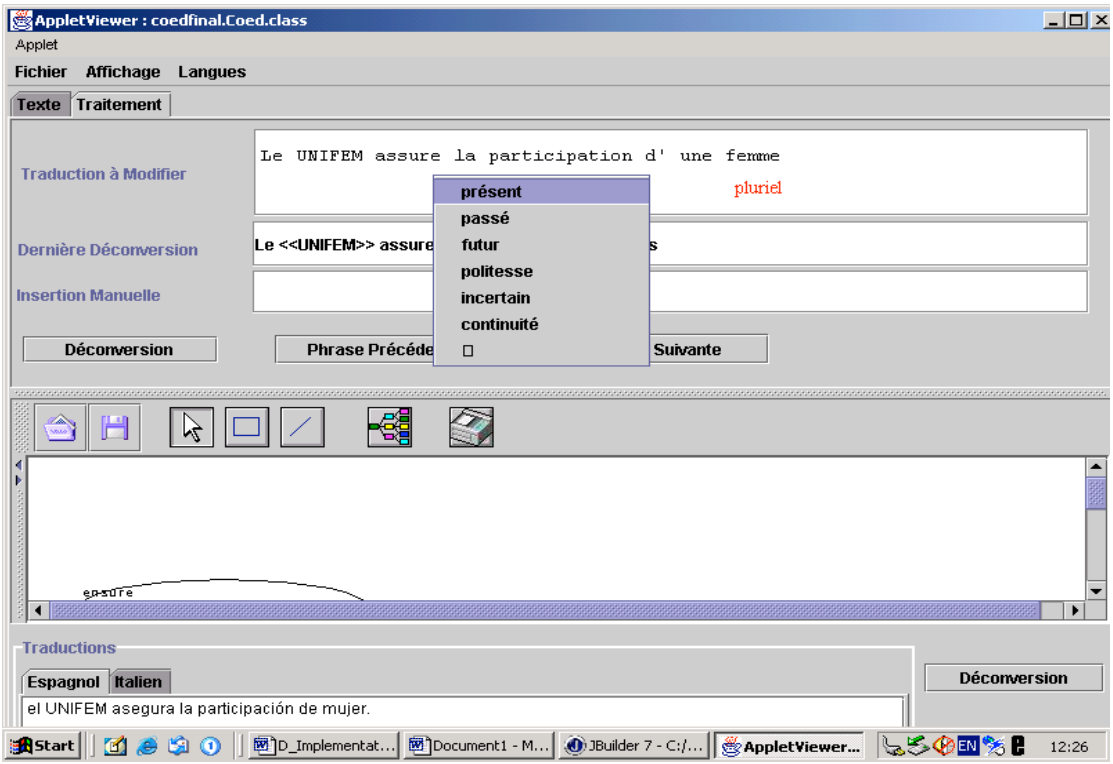


Fig. D-38 Propositions pour modifier un verbe

Enfin, l'utilisateur retourne au mode de lecture et le visualisateur affiche le nouveau texte déconverti à partir du graphe UNL coédité (le groupe « d'une femme » a été changé en « des femmes »).

Comme le déconvertisseur n'a pas généré l'élision correcte (Le UNIFEM), l'utilisateur termine de corriger dans la zone manuelle. Dans cet exemple, il retourne au mode de lecture sans avoir coédité les deux autres phrases. On obtient alors en français :

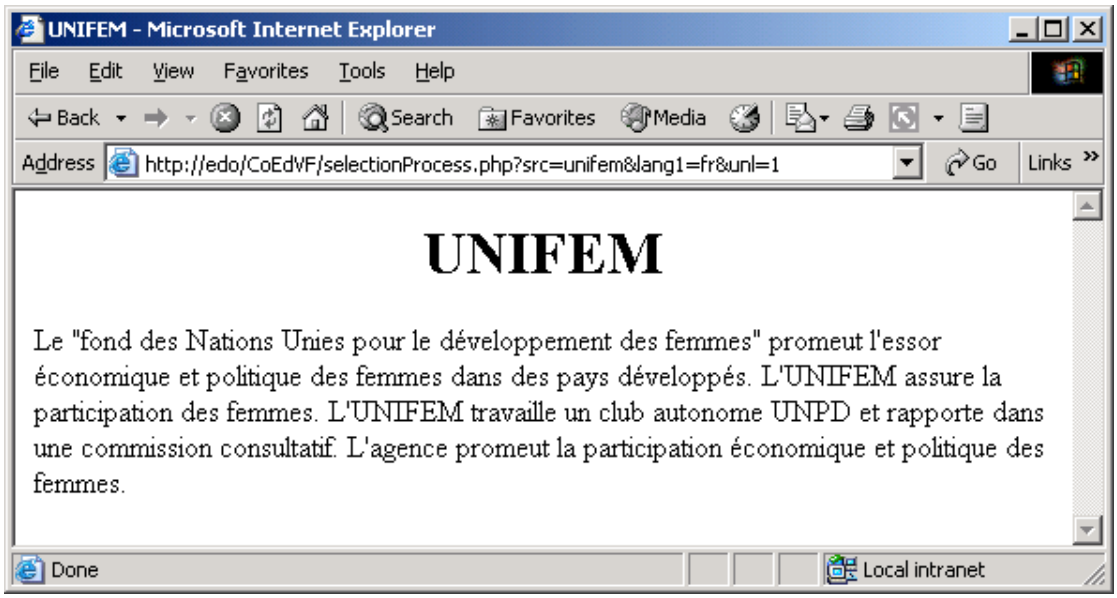


Fig. D-39 Lecture de nouveau texte

Avant cette opération de coédition, la version espagnole apparaissait comme dans les Fig. D-36 ou Fig. D-37 (El UNIFEM asegura la participación de mujer). Après, si on déconvertit de nouveau vers l'espagnol, on obtient la figure suivante (El UNIFEM asegura la participación de mujeres).

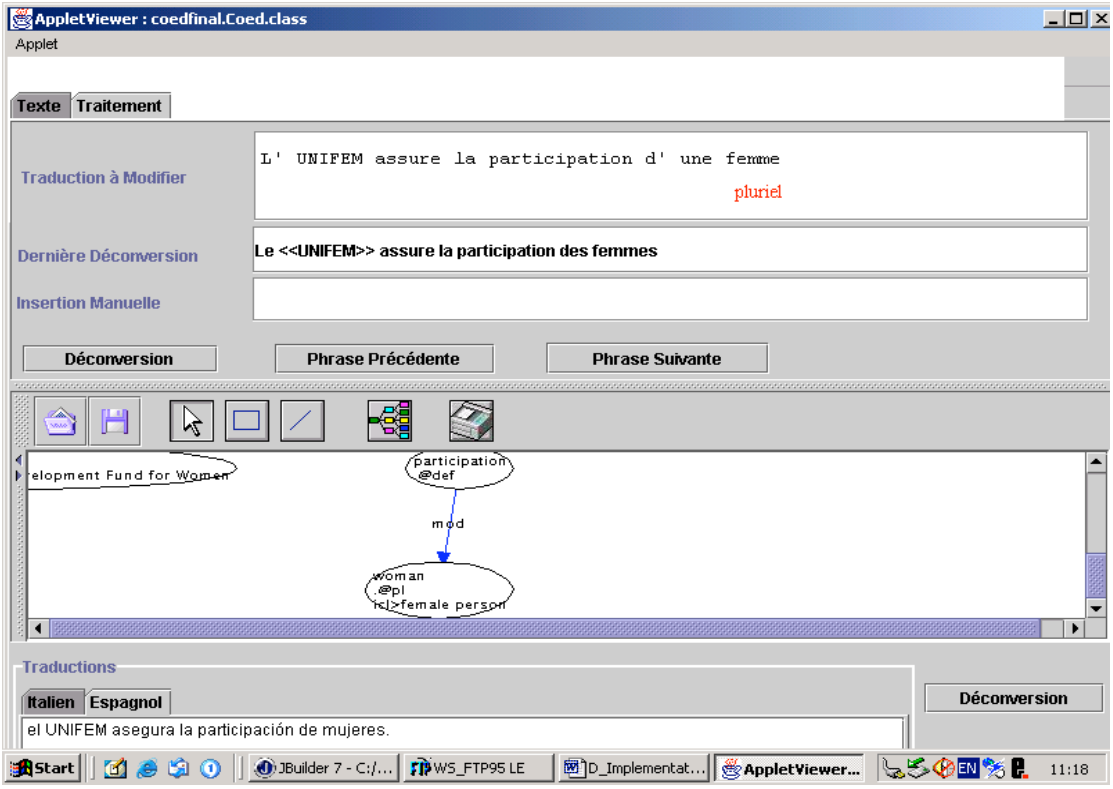


Fig. D-40 Déconversion vers l'espagnol

On voit que la correction a bien été transportée du français vers l'espagnol, puisqu'on avait « mujer » et qu'on a maintenant « mujeres ». Et comme il n'y a pas en espagnol de problème lié au déconvertisseur (comme « Le UNIFEM »), la phrase corrigée est parfaite, il n'y a aucune correction manuelle à faire.

Voici les versions italiennes et russes.

Avant la coédition :

UNIFEM _____ _____ _____.

UNIFEM garantisce che participation di donna .

Après la nouvelle déconversion :

UNIFEM _____ _____ _____.

UNIFEM garantisce che participation di donne.

La nouvelle déconversion russe a bien tenu compte de la correction (« _____ » féminin, singulier, génitif _ « _____ » , féminin, pluriel, génitif). Les phrases italiennes sont erronées mais la modification y a bien été transportée.

2.2.3 Architecture interne et classes principales

Voici une figure qui donne une vision générale de l'ensemble de la maquette :

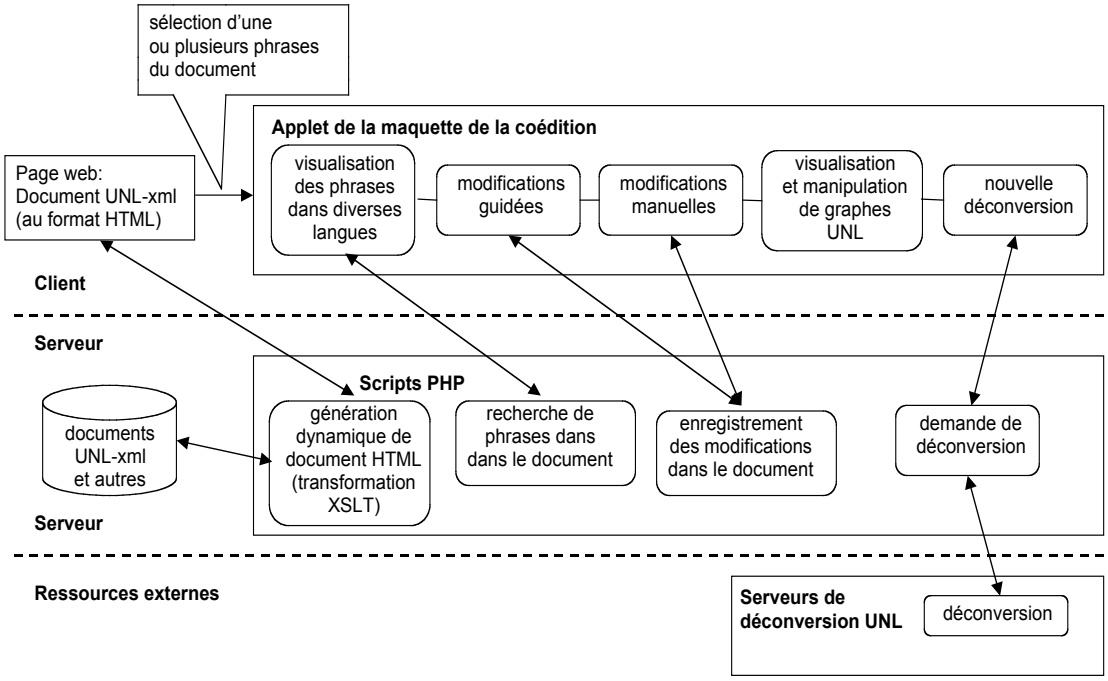


Fig. D-41 Vue générale de la maquette

Voici une liste des fonctionnalités réalisées :

choix d'un texte dans une langue donnée,

lecture du texte,

choix des langues de lecture,

sélection des phrases (contenant la sélection courante),

choix d'une phrase à éditer,

modification du graphe,

entrée manuelle du texte,

choix de la langue d'interface,

propositions d'une liste de modifications,

annotation du texte par la modification choisie et exécution de l'action correspondance sur le graphe,

déconversion vers le français.

Voici les classes construites dans cette maquette :

Graphe – classe permettant de représenter un graphe UNL. Elle comprend une table de hâchage des Nœuds et une table de hâchage des Relations.

Relation – classe permettant de représenter une relation du graphe UNL.

Nœud – classe permettant de représenter un nœud du graphe UNL.

Phrase – classe permettant d'extraire une phrase d'un document UNL-xml et de la transformer en HTML afin de conserver les informations de pré-traitement par un système de balises HTML de type « span ».

Texte – classe permettant de gérer l'extrait de document UNL-xml au niveau de l'applet.

HTTPMessage – classe représentant un message http envoyé par la servlet au serveur.

traitementXML – classe permettant d'effectuer divers traitements DOM XML sur les fichiers UNL-xml.

HTTPPost – classe permettant l'envoi de requêtes POST.

L'affichage du graphe (dans un onglet) a été développé en exploitant une bibliothèque Java gratuite, développée par M. Jesus Salvo (openjgraph.sourceforge.net) et la connexion entre l'applet et les déconvertisseurs est écrite en PHP [Bernhard 02].

Pour construire la correspondance entre la chaîne et l'arbre, on a créé trois classes Java :

CorresMot –Classe permettant de stocker les informations lexicales d'un mot donné.

CorresNoeud – Classes permettant de stocker les informations relatives à un nœud donné.

Correspondance – Classes permettant d'établir la correspondance entre texte et graphe.

La construction de la correspondance se réalise par la classe « Correspondance ». Ses deux données membres de base (reçues en paramètre par son constructeur) sont *phraseLN*, la chaîne de la phrase en langue naturelle, et *graphe*, un objet Graphe décrivant le graphe UNL correspondant.

En premier lieu, Correspondance segmente phraseLN en mots, et crée à partir de chacun d'eux un CorresMot, qui sera stocké dans un vecteur corresMots. Elle leur passe en paramètre la chaîne *pilaf*, résultat de l'analyse morphosyntaxique de *phraseLN*, ce qui leur permet de remplir leur vecteur *infoLemme*.

De la même façon, elle passe en revue les nœuds de *graphe* et crée pour chacun un CorresNoeud qui sera stocké dans un vecteur corresNoeuds.

C'est également au niveau de l'objet Correspondance que se fait l'appel au dictionnaire pour trouver les traductions des lemmes et des noms de nœuds. Le résultat de la recherche est stocké dans la chaîne *dico*, puis redistribué aux CorresMot et aux CorresGraphe, lesquels prélèvent les traductions qui les concernent.

On voit notamment qu'un objet Correspondance est composé d'au moins un objet CorresMot et d'au moins un objet CorresNoeud. Un objet CorresMot peut correspondre à la totalité ou à une partie d'un objet CorresNoeud.

La classe Correspondance comporte une méthode *match()*, qui parcourt les CorresMot et essaie de trouver le CorresNoeud correspondant. Un vecteur *lien* associe un

CorresMot, un CorresNoeud, et un type de correspondance. Une autre fonctionnalité de *match()* est qu'il nous permet de remplir des patrons spécifiés pour trouver d'autres correspondances que lemme-UW. Par exemple, « detp-.@def ».

2.2.4 Évaluation et points à améliorer dans la version _ de la maquette

La version _ que nous venons de présenter est une version réduite de ce que nous avons présenté en partie C, par rapport à la structure de données et à l'algorithme.

Par rapport à l'algorithme que nous avons proposé dans la partie C, cette maquette n'établit en effet que des liaisons lexicales. Quant à l'enrichissement des liaisons, elle ne traite que quelques patrons, notamment, « detp-.@def », « pl-.@pl », et les quatre attributs pour les quatre catégories lexicales (nom, verbe, adjectif, adverbe).

Dans cette version _, il reste les points suivants à améliorer :

Compléter l'implémentation de notre algorithme – Lors des stages pendant lesquels cette version _ a été implémentée, notre algorithme n'était pas assez mûr. La structure de données utilisée dans cette version est trop simple pour réaliser le calcul de tous les types de correspondances.

Traiter des graphes UNL arbitraires - les graphes que le programme traite maintenant doivent être des graphes arborescents (c'est la limitation du module de l'affichage graphique). La maquette ne peut pas traiter non plus un graphe avec scope.

Se connecter à plusieurs déconvertisseurs (cette version ne se connecte qu'au serveur français).

Synchroniser les deux fenêtres montrant le graphe et le texte.

Ajouter une servlet java ou réécrire l'applet en une servlet côté serveur pour sauvegarder le résultat de coédition.

L'environnement de développement Java était Jbuilder, qui comprend sa propre bibliothèque et cela réduit la portabilité de la maquette, qui ne fonctionne pas sur toutes les plates-formes. Dans la suite, on a utilisé les bibliothèques du JDK.

2.2.5 Quelques mots sur la proposition de correction

Au cours du développement de la maquette, nous nous sommes rendu compte que la réalisation d'un menu proposant aux utilisateurs des modifications possibles est une tâche compliquée. En effet, elle est liée à plusieurs facteurs :

les liaisons que système est capable de créer,

les caractéristiques d'UNL qui peuvent être reflétées sur la langue naturelle,

les caractéristiques de la langue naturelle traitée,

les modules AMS exploités et les résultats qu'ils produisent,

les types de correction que les utilisateurs peuvent comprendre,

les types de correction que les utilisateurs peuvent faire par un seul clic sur un mot,

les types d'erreurs corrigibles (nous avons discuté ce point en section B.3.2.4),

les rapports entre les modifications (l'exclusion ou l'accord, par exemple) et la portée d'une modification.

Les propositions présentées dans le menu doivent être assez abstraites, donc dans une sorte de métalangage, pour que le système n'ait pas besoin de faire des calculs supplémentaires au moment de produire ce menu. Mais il faut pas être trop abstrait ou trop académique, puisque nous visons les utilisateurs non-spécialistes.

Ainsi, dans la conception originelle de cette maquette _, les propositions faites par le système n'étaient pas bonnes, parce qu'elles consistaient en des formes complètes. C'est plus facile à comprendre pour les utilisateurs, mais le système doit calculer toutes ces formes (déclinaisons et conjugaisons régulières et irrégulières), ce qui n'est pas possible sans générateur morphologique pour une langue riche en déclinaisons comme le français.

C'est pourquoi, au lieu de proposer les formes « femme » ou « femmes », la version _ actuelle propose « singulier » ou « pluriel », comme nous l'avons montré dans la Fig. D-37.

D'autre part, le verbe français est très compliqué, riche en conjugaisons. Le mode subjonctif peut correspondre à plusieurs attributs possibles d'UNL. L'aspect n'est pas très clair pour les Français. Ce sont des cas qui méritent plus de réflexion avant de proposer aux utilisateurs de les modifier. Nous avons déjà montré une tentative de propositions dans la Fig. D-38.

Même pour des langues proches du français, comme l'espagnol, les menus de coédition ne peuvent pas être identiques. Nous donnons la phrase suivante comme exemple :

UNL : agt(come(icl>do, agt>human).@entry.@present, they(icl>man))
tim(come(icl>do, agt>human).@entry.@present, today)

français : « ils viennent aujourd'hui »

espagnol : « vienen hoy »

Il faut alors que le menu de coédition en espagnol ne contienne pas le nombre sur « vienen », sauf si on établit une liaison entre la variable grammaticale (troisième personne, pluriel) et « they(icl>human) », mais cela est trop difficile. En principe, en espagnol le mot « vienen » n'est pas coéditable pour le pluriel, puisqu'un nœud UNL contenant une UW verbale n'a pas normalement d'attribut de nombre. Par contre, en français, on aura « ils/elles viennent », donc il est coéditable : ils_they(icl>man) ou elles_they(icl>woman). Il est possible de proposer aux utilisateurs de changer le sujet ils/elles en français. On entrevoit des solutions, comme d'insérer un pronom sujet (facultatif), comme « [ellos] vienen », mais ce serait très ad hoc et sans doute délicat à implémenter. Il faudrait, par exemple, mettre « [por ellos] » à la bonne place s'il s'agit d'un verbe transitif ou passif.

En bref, l'étude des modifications proposables n'est pas simple et la conception différera vraiment selon la langue en question.

2.2.6 Nouvelle maquette

En ce moment, nous sommes en train de programmer une nouvelle maquette de coédition, dans laquelle nous implémenterons notre algorithme complet (ajout de liaison non lexicales pour l'établissement de l meilleure correspondance arbre UNL-treillis LMS) et améliorerons les points faibles constatés.

Pour cela, nous partons des programmes du serveur UNL-français réalisé par G. Sérasset en Java sous Enhydra [Endydra] qui tourne actuellement au GETA (sous Linux). Ce serveur est destiné au public ainsi qu'aux développeurs UNL. Le public y accède par une page web (http://gohan.imag.fr:8002/), et les développeurs peuvent y accéder par une requête CGI.

Il y a deux fonctionnalités principales pour les développeurs : la déconversion et la gestion du dictionnaire français-UNL. Sous la page de déconversion, on trouve 5 fonctionnalités pour mieux déboguer le graphe UNL :

afficher le graphe,

afficher l'arbre (ARIANE, avec la transformation lexicale),

éditer l'arbre et régénérer l'arbre Ariane,

éditer le graphe,

déconvertir le graphe.

Voici la page web de ce serveur.

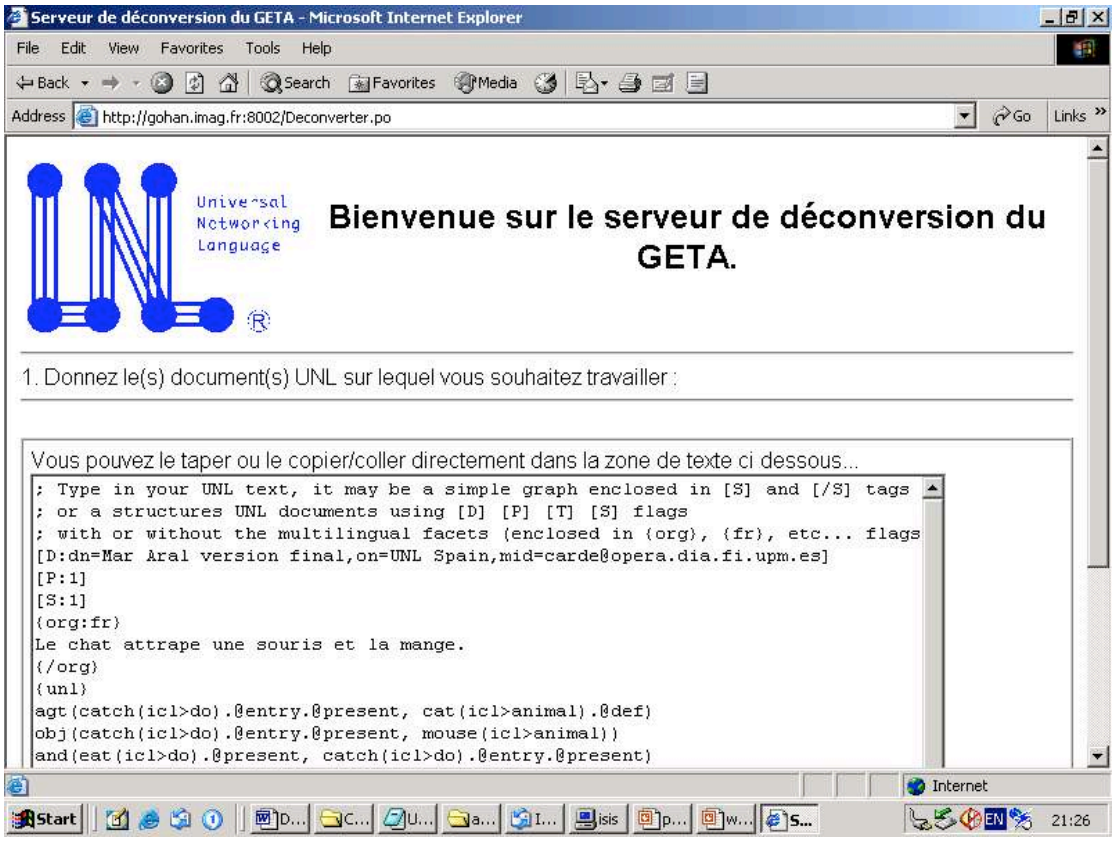


Fig. D-42 Page web principale du serveur de déconvertisseur UNL-français

Les limites de cette nouvelle maquette sont les suivantes :

une seule langue de coédition (français), et un seul AMS (PILAF),

un seul dictionnaire, à savoir le dictionnaire UW-FR (réversible),

limitations à certaines possibilités de coédition : sur les noms (nombre, sexe, détermination), sur les verbes (temps abstrait, négation, politesse, aspect), sur la négation (ni, ne… pas), et sur les articles, pronoms et articles-pronoms (catégorie nomp de PILAF : nombre, personne, sexe).

Voici une liste des groupes de classes Java que nous réutilisons entièrement ou partiellement :

unl-graph : pour traiter le graphe UNL,

ariane : pour construire un arbre ARIANE,

dictionary : pour consulter le dictionnaire (dans une base de données Postgres),

deconverter : pour gérer l'interface du déconvertisseur,

lidia : pour contacter le serveur LIDIA (sur un IBM-H30, sous VM/CMS), où tourne la partie linguistique du déconvertisseur français.

D'autre part, nous réutilisons certains modules de la maquette version β :

la lecture d'un document UNL-xml,

le passage d'une page web à la maquette de coédition (sélection d'un nombre entier de phrases, etc.)

Enfin, nous réécrirons sous Enhydra toutes les classes gérant le texte et le treillis, notre interface de coédition, les fichiers UNL-xml, et aussi les classes permettant de contacter PILAF et les autres déconvertisseurs.

Voici une vue générale de cette nouvelle maquette :

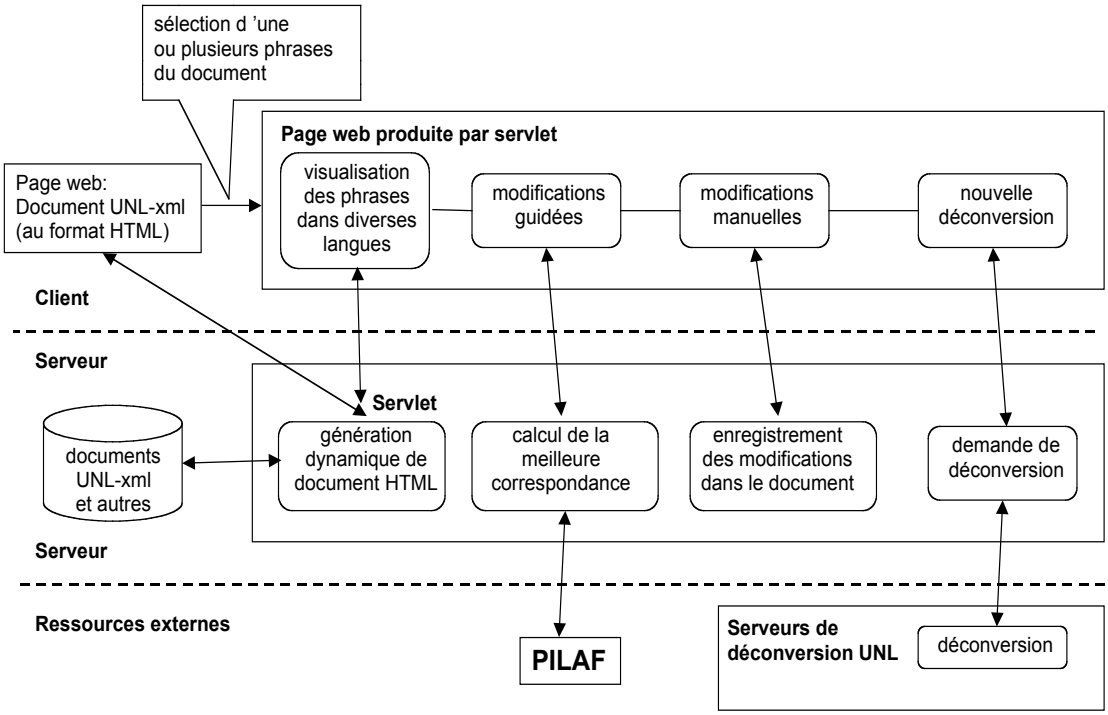


Fig. D-43 Vue générale de la nouvelle maquette

3. Bilan et conclusion

3.1 Amélioration dans la nouvelle déconversion

Nous montrons ici deux exemples de l'amélioration de la qualité de déconversion par coédition.

Le premier a été fait pour une démonstration dans le cadre du projet « la main à la pâte », pour montrer que l'amélioration du graphe UNL peut se propager dans les autres langues. Les déconversions de l'italien, de l'espagnolm du russe ou du chinois ont été effectivement produites par les déconvertisseurs et non révisées. Le graphe UNL est le suivant (étape 0) :

{org :fr}Des enfants regardent un phénomène du monde vrai proche et perceptible.{/org}
{unl}
mod:01(phenomenon.@entry,world.@def)
mod:01(world.@def,real)
mod:01(world.@def,perceptible)
and:01(perceptible,near(mod<thing))
obj(watch(icl>do).@entry,:01)
agt(watch(icl>do).@entry,child.@pl)
{/unl}
{fr}Des enfants regardent un phénomène du monde véritable proche et sensible. {/fr}
{it} Bambini guardare fenomeno del vero il mondo percettibile. {/it}
{es} Niño miran a el fenómeno del mundo real de cercano y perceptible. {/es}
{cn}_____{/cn}

{ru}____ _____ _____ _____ _ _____ _____ ____.
{/ru}

La liaison entre « enfant » et « child » peut être facilement créée. Supposons qu'on choisit « défini » pour le nom, ce qui a pour effet d'ajouter « .@def » dans le nœud « child » du graphe, puis qu'on demande une nouvelle déconversion quadrilingue. Voici les résultats (étape 1).

{org :fr}Des enfants regardent un phénomène du monde vrai proche et perceptible.{/org}
{unl}
mod:01(phenomenon.@entry,world.@def)
mod:01(world.@def,real)
mod:01(world.@def,perceptible)
and:01(perceptible,near(mod<thing))
obj(watch(icl>do).@entry,:01)
agt(watch(icl>do).@entry,child.@pl.@def)
{/unl}
{fr}Les enfants regardent un phénomène du monde véritable proche et sensible. {/fr}
{it} I bambini guardare fenomeno del vero il mondo percettibile. {/it}
{es} Los niño miran a el fenómeno del mundo real de cercano y perceptible. {/es}
{cn}_____{/cn}
{ru}____ _____ _____ _____ _ _____ _____ ____.
{/ru}

Remarquons que le chinois et le russe n'ont pas été modifiés, parce que la détermination ne s'exprime pas toujours en chinois et en russe.

Puis on met la négation sur le verbe « regardent », c'est-à-dire qu'on ajoute « .@not » dans le nœud « watch » du graphe. Voici les résultats (étape 2) :

{org :fr}Des enfants regardent un phénomène du monde vrai proche et perceptible.{/org}
{unl}
mod:01(phenomenon.@entry,world.@def)
mod:01(world.@def,real)
mod:01(world.@def,perceptible)
and:01(perceptible,near(mod<thing))
obj(watch(icl>do).@entry.@not,:01)
agt(watch(icl>do).@entry.@not,child.@pl.@def)
{/unl}
{fr}Les enfants ne regardent pas un phénomène du monde véritable proche et sensible. {/fr}
{it} I bambini non guardare fenomeno del vero il mondo percettibile. {/it}
{es} Los niño no miran a el fenómeno del mundo real de cercano y perceptible. {/es}
{cn}_____{/cn}
{ru}____ __ _____ _____ _____ _ _____ _____ ____.
{/ru}

Voici un tableau ne montre que les fragments textuels avant à chaque étape.

étape	français	italien	espagnol	chinois	russe
0	Des enfants regardent un phénomène du monde véritable proche et sensible.	Bambini guardare fenomeno del vero il mondo percettibile.	Niño miran a el fenómeno del mundo real de cercano y perceptible.	_____ _____ ___	____ _____ _____ _____ _ _____ _____ ____.
1 (.@def)	Les enfants regardent un phénomène du monde véritable proche et sensible.	I bambini guardare fenomeno del vero il mondo percettibile.	Los niño miran a el fenómeno del mundo real de cercano y perceptible.	_____ _____ ___	____ _____ _____ _____ _ _____ _____ ____.
2 (.@not)	Les enfants ne regardent pas un phénomène du monde véritable proche et sensible.	I bambini non guardare fenomeno del vero il mondo percettibile.	Los niño no miran a el fenómeno del mundo real de cercano y perceptible.	____ __ _____ ____	____ __ _____ _____ _____ _ _____ _____ ____.

Tableau D-3 Propagation de modifications

Par ces deux modifications, nous pouvons avoir une idée de la propagation de modifications à partir d'une langue naturelle, à travers le graphe UNL, vers d'autres langues. Ces deux exemples montrent deux choses :

> Certaines modifications ne s'expriment pas dans certaines langues, quand elles ne sont pas importantes dans ces langues (mais c'est très souvent l'origine de la mauvaise traduction ou de l'ambiguïté entre langues).
>
> Dans les modifications simples comme celles que nous avons montrées, la coédition peut avoir un résultat remarquable, surtout entre langues proches.

Le deuxième exemple est le résultat de la construction du petit corpus « la main à la pâte ». Chaque équipe a ajusté son déconvertisseur et complété son dictionnaire UNL-LN pour ce corpus. Il y a donc deux versions de déconversion pour chaque équipe. Voici un extrait de ce corpus :

> {org:fr} De nombreuses ressources (pédagogiques, scientifiques, activités de classe) ainsi que des outils d'échange y sont proposés{/org}
> {unl} obj(offer(icl>do).@entry,tool(icl>thing).@topic.@pl)
> mod(tool(icl>thing).@topic.@pl,exchange(icl>abstract thing))
> and(tool(icl>thing).@topic.@pl,resource:01.@pl)
> mod(resource:01.@pl,many) mod(resource:01.@pl,:01.@parenthesis)
> mod:01(activity(icl>abstract thing).@entry.@pl,class(icl>school))
> and:01(activity(icl>abstract thing).@entry.@pl,resource:02.@pl)
> mod:01(resource:02.@pl,scientific(mod<thing))
> and:01(resource:02.@pl,resource:03.@pl)
> mod:01(resource:03.@pl,pedagogical(mod<thing){/unl}

{fr:01} De nombreuses ressources (de ressources <<PEDAGOGICAL>> de ressources scientifiques et d'activités d'une classe) et des outils d'un <troc> sont offertes{/fr}
{fr:02} Beaucoup de ressources (de ressources pédagogiques de ressources scientifiques et d'activités d'une classe) et des outils d'un échange sont offertes.{/fr}
{it:01} strumenti e molte risorse (di attivita' , di risorse e di risorse pedagogiche scientifiche di classe)che scambiano offrire {/it}
{it:02} strumenti di scambio e molte risorse (di attivita' , di risorse e di risorse pedagogiche scientifiche di classe)sono offerti {/it}
{es:01} muchos recursos de recursos de pedagogical(mod<thing), recursos científicos y actividades de clasificaban y herramientas de intercambiar son brindada.{/es}
{es:02} muchos recursos de recursos pedagógicos, recursos científicos y actividades de clase y herramientas de intercambio son brindadas.{/es}
{ru:01} _____ _____ _____ _____, _____ _____ _ (_____ _____) _ _____ _____ _____.{/ru}
{ru:02} _____ _____ _____ _____ _____, _____ _____ _ (_____ _ _____) _ _____ _____.{/ru}

Remarquons l'amélioration apportée dans chaque langue :

français : dans la première version, le mot « pédagogique » n'était pas dans le dictionnaire, et le déconvertisseur a choisi « troc » pour « exchange ». Tout est corrigé après deux petites modifications dans le dictionnaire. On peut remarquer que, même dans la deuxième déconversion, la partie « des outils d'un échange » est erronée. Clairement, il manque en UNL l'attribut « @abs » pour l'emploi absolu.

italien : la syntaxe a été améliorée dans la deuxième version. De plus, dans la première version, « exchange » a été déconverti comme un verbe (« scambiare »). Dans la deuxième déconversion, cela a été corrigé.

espagnol : dans la première version, « pedagógicos » n'était pas dans le dictionnaire, donc l'UW est sortie telle quelle. Le nom « classe » avait été déconverti en un verbe, « clasificar », ce qui est faux.

russe : la première version utilisait un verbe réfléchi (« il est proposé ») et la deuxième utilise un verbe non réfléchi (« on propose »). La première version a déconverti « activité de classe » en utilisant la relation sémantique « possessif », la deuxième version a correctement déconverti la relation sémantique « lieu » entre « activité » et « classe ».

Comme nous l'avons constaté dans la section B.3.2.4, il y a des erreurs non corrigibles par la coédition, mais il est important de montrer les possibilités aussi bien que les limitations de la coédition. Du côté organisationnel, notons que l'approche collaborative du projet a très bien fonctionné pour cette expérience, tant que au niveau des grammaires de déconversion qu'à celui des dictionnaires UNL-Lg.

3.2 Conclusion

Notre maquette « _ » nous a permis de progresser, mais elle suppose que les documents UNL sont mis dans un format xml beaucoup plus détaillé que le format

UNL-xml « de base ». D'autre part, elle ne permet pas de gérer le document UNL-xml « maître » sur le serveur. Par conséquent, on ne peut pas l'utiliser pour faire des expériences avec des utilisateurs distants, ni en dériver un prototype opérationnel.

C'est pourquoi nous avons commencé à développer la première version d'un système de lecture et de coédition par le web de documents UNL-xml, ou de documents UNL-html, en réutilisant l'architecture et un grand nombre des programmes du serveur de déconversion UNL-FR écrit en Java sous Enhydra par G. Sérasset.

Nous arrivons maintenant à la fin de cette dernière partie, consacrée à diverses implémentations réalisées pour progresser vers le raffinement du paradigme de coédition, et vers sa réalisation, qui semble maintenant proche, sous une forme opérationnelle.

Nous avons spécifié le cahier des charges et les scénarios types pour la plate-forme d'expérimentation SWIIVRE-UNL, puis décrit l'évolution et l'état courant de ce site web. La réalisation de la plate-forme SWIIVRE-UNL nous a donné une vue plus générale sur les applications d'UNL et sur l'exploitation d'UNL sur Internet. Nous avons réalisé plusieurs modules, classés en six catégories principales, et les avons mis en service sur cette plate-forme.

La réalisation de la maquette version _ a été une première expérience, qui nous a permis d'avancer dans la conception de l'interface, et dans la conception du menu de propositions. Elle a donné un résultat assez encourageant, prouvant qu'on peut effectivement créer des liaisons entre le graphe UNL et le texte, bien que le graphe traité soit encore simple.

Enfin, nous avons brièvement présenté les principes d'implémentation d'une nouvelle maquette, plus complète, qui pourrait être la base d'un premier service expérimental, et devrait pouvoir être présentée prochainement.

Conclusion

Rappel de la situation et du problème

Ce que nous aimerions, c'est produire des documents multilingues d'abord par TA, et faire en sorte que le travail de révision puisse être partagé entre les langues, quels que soient le domaine et le contexte.

Nos trois idées principales sont :

Mutualisation et collaboration,

Révision à la demande,

Partage de révision parmi les différentes langues.

Les problèmes les plus importants à résoudre pour la mise en œuvre de ces idées sont :

quelle « structure intermédiaire » choisir ?

comment faire modifier une structure intermédiaire de ce genre par des utilisateurs « naïfs » ?

comment établir une correspondance fine entre le texte et la structure ?

Apports de cette thèse

Au niveau de la conception, nous avons commencé notre recherche par une étude des système de « coédition » disponibles et nous avons proposé une taxonomie des types de « coédition », puis nous avons identifié les caractéristiques souhaitables dans ces systèmes de coédition. Nous avons discuté de la façon d'adapter l'idée de coédition à la communication multilingue écrite.

Ayant décidé d'exploiter une structure pivot, nous avons continué en cherchant le langage pivot le plus adéquat. Nous avons étudié huit systèmes de TA utilisant un langage pivot, et en avons déduit les caractéristiques du langage pivot que nous souhaitons pour notre système. Enfin, nous avons décidé d'utiliser UNL comme langage pivot. Nous avons ensuite fait une étude complète du système UNL, conçu des scénarios pour un système de coédition fondé sur UNL, et identifié les types d'erreurs corrigibles par la coédition.

Au niveau de la théorie, nous avons fait une étude de la formalisation de correspondances entre structures différentes. Nous avons aussi fait une étude sur la correspondance entre graphes UNL et éléments d'énoncés en diverses langues naturelles. Cette étude a été menée sur les corpus UNL. Nous l'avons exploitée pour proposer ensuite un algorithme heuristique pour trouver la meilleure correspondance entre un texte et un graphe UNL.

Au niveau de l'implémentation, nous avons construit une plate-forme pour l'expérimentation d'UNL, et plusieurs modules liés à l'usage d'UNL sur Internet. Nous avons proposé un format « UNL-xml » pour faciliter le traitement et l'échange

de documents UNL sur Internet, et nous avons aussi construit des outils autour de ce format. Enfin, nous avons réalisé une maquette version β pour montrer les scénarios et l'environnement de coédition. Nous travaillons maintenant sur une nouvelle maquette.

Nous avons aussi montré quelques résultats et plusieurs façons d'améliorer un texte multilingue, par la coédition et par la mise à jour de dictionnaires et de déconvertisseurs. Le résultat est plutôt prometteur. Selon l'expérience du GETA, au moins 60% de la révision pour la TA russe-français concerne des fautes d'articles, la détermination, ce qui peut être corrigé par la coédition.

En conclusion, nos travaux montrent que, sans construire des outils ou des modules compliqués et coûteux, mais en utilisant des modules existants et gratuits, nous pouvons construire des liaisons à travers une (des) structure(s) intermédiaire(s) entre le texte et le graphe UNL correspondant. Cela permettra aux utilisateurs de modifier les textes et corriger certains types d'erreur dans leur langue, et de propager ces corrections vers les autres langues.

L'idée de la coédition est neuve, sa puissance est sa simplicité. Nous ne prétendons pas pouvoir corriger tous les types d'erreur, ni établir une correspondance parfaite, car cela ne serait pas le cas non plus avec des outils ou modules puissants et coûteux.

Perspectives de recherche

Quand nous disposerons d'une version réellement opérationnelle sur le web de notre système, nous pourrons évaluer la coédition, selon les axes suivants :

« couverture » moyenne des correspondances graphe-texte, i.e. pourcentage d'éléments liés des deux côtés,

influence des poids des patrons sur la couverture des correspondances trouvées,

généricité, en appliquant la coédition à une langue très différente du français.

Quant au système de coédition lui-même, voici trois extensions possibles :

désambiguïsation interactive au niveau de la correspondance texte-treillis AMS : comme le fait Systran dans sa toute dernière version 5, on pourrait proposer à l'utilisateur de corriger le choix automatique en modifiant la trajectoire sélectionnée dans le treillis.

édition du graphe UNL par manipulation directe, soit tel quel, soit dans une présentation localisée dans la langue de l'utilisateur, comme le fait l'équipe UNL espagnole.

généricité au niveau de la langue de coédition, de l'AMS utilisé, et des ressources lexicales. En fait, nous avons limité notre maquette à la langue française et à l'AMS PILAF. Il sera intéressant de concevoir un système permettant aux utilisateurs de changer la langue de coédition, les dictionnaires et les déconvertisseurs à utiliser.

Enfin, il y a encore une question intéressante :

Supposons qu'on ait un document UNL contenant une version dans une langue non liée à UNL, ainsi que les graphes UNL. On pourrait coéditer le graphe UNL d'une

phrase, en établissant des correspondances texte-graphe, puisque cela ne suppose ni déconvertisseur ni enconvertisseur. On pourrait ainsi améliorer les version de cette phrase dans les « langues UNL » (ayant un déconvertisseur), mais pas dans cette langue L. La question est alors de savoir si on pourrait « apprendre » un déconvertisseur et un enconvertisseur de L à partir d'un grand corpus L-UNL, les phrases en langue L ayant été obtenues par une méthode de traduction quelconque ne faisant pas intervenir UNL.

Bibliographie

[Abney 96] Steven Abney, *Part-of-Speech Tagging and Partial Parsing*. In: Church, Ken; Young, Steve; Bloothooft, Gerrit(eds.) *Corpus-Based Methods in Language and Speech.* Dordrecht: Kluwer Academic Publishers. 1996. 25 p.

[Al-Adhaileh 98] Al-Adhaileh M. H., Tang E. K., *A Flexible Example-Based Parser Based on the SSTC***.** Proceedings of the 17th International Conference on Computational Linguistics (COLING'98), Montreal, Canada, August 1998. pp. 687-693.

[Al-Adhaileh 99] Al-Adhaileh M. H., Tang E. K., *Example-Based Machine Translation Based on the Synchronous SSTC Annotation Schema.* Proceedings of Machine Translation Summit VII 99, Singapore, September 1999. 10p.

[Al-Adhaileh 02a] Al-Adhaileh M. H., Tang E. K., *Synchronous Structured String-Tree Correspondence (S-SSTC).* The 20th IASTED02 International Conference, Innsbruck, Austria. 2002. 6p.

[Al-Adhaileh 02b] Al-Adhaileh Mosleh Hmond, Synchronous Structured String-Tree Correspondence (S-SSTC) and its applications for machine translation. Ph. D. Thesis, UTMK, Universiti Sains Malaysia, 2003. 163p.

[Al Assimi 00] Al Assimi A.-B., *Gestion de l'évolution non centralisée de documents parallèles multilingues.* Nouvelle thèse, UJF, Grenoble, 31/10/00. 200 p.

[Al Assimi 01] Al Assimi A.-B. & Boitet Ch., *Management of Non-Centralized Evolution of Parallel Multilingual Documents.* Proc. Internationalization Track, 10th International World Wide Web Conference, Hong Kong, May 1-5, 2001. 7 p.

[Bernhard 02] Bernhard D., Echinard S., Helme S., *Système de post-édition utilisant UNL*. Rapport du projet génie logiciel de DESS Double Compétence Informatique et Sciences Sociales Université Pierre Mendès-France, mai-juin 2002. 73 p.

[Besacier 01] Besacier L., Blanchon H., Fouquet Y., Guilbaud J.-P., Helme S., Mazenot S., Moraru D. and Vaufreydaz D., *Speech Translation for French in the NESPOLE! European Project.* Proc. Eurospeech. Aalborg, Denmark. September 3-7, 2001. pp. 1291-1294.

[Bey 03] Youcef Bey, *Génération aléatoire de corpus et calcul de relations de dépendance avec apprentissage,* Rapport de DEA d'informatique :systèmes et communications, Université Joseph Fourier, 23/06/2003, 61p.

[Blanc 00] Blanc E., *From the UNL hypergraph to GETA's multilevel tree*, MT2000 Conference, Exeter, UK, 20-22/11/2000.

[Blanc 01] Blanc E. & Sérasset G., *From Graph to Tree: Processing UNL Graphs using an Existing MT system*, First International UNL Open Conference, Suzhou, China, 26-29/11/2001. 9 p.

[Blanc 02] Blanc E., *Structural and Lexical Transfer From an UNL graph to an Equivalent NL Dependent Tree*, Workshop "First International Workshop on UNL, other Interlinguas and their Applications", LREC2002, Las Palmas, Spain, 27/5-2/6/2002. pp. 14-18.

[Blanchon 94] Blanchon H., *LIDIA-1: Une première maquette vers la TA interactive « pour tous »*. Nouvelle thèse, UJF, Grenoble, 21/01/94. 321p.

[Boitet 76] Boitet Ch., Un essai de réponse à quelques questions théoriques et pratiques liées à la traduction automatique définition d'un système prototype. Thèse d'Etat, 16/04/1976. 219 p.

[Boitet 78] Boitet Ch., Guillaume P., Quézel-Ambrunaz M, *Manupulation d'arborescenes et parallélisme : le système ROBRA*. Proc. COLING-78, Bergen, Qugust 1978, 12p.

[Boitet 86] Boitet Ch., *Software and Lingware Engineering in modern M(A)T Systems*. Handbook of Machine Translation (Niemeyer 1987). 15 p.

[Boitet 88a] Boitet Ch. & Zaharin Y., *Representation trees and string-tree correspondences*. Proc. COLING-88, Budapest, 22–27 Aug. 1988. pp. 59-64.

[Boitet 88b] Boitet, Ch., *Pros and Cons of the Pivot and Transfer Approach in Multilingual Machine Translation*, Dan Maxwell, Klaus Schubert & Toon Witkam (editors), New Directions in Machine Translation, Dordrecht, Foris. pp. 93-106.

[Boitet 88c] Boitet, Ch., *Bernard VAUQUOIS' contribution to the theory and practice of building MT systems : a historical perspective*, Second int. conf. on theoretical and methodological issues in the machine translation of natural languages, Pittsburgh, June, 1988. 18 p.

[Boitet 88d] Boitet Ch., *Hybrid Pivots Using m-Structures for Multilingual Transfer-Based MT Systems*. In the Meeting of Japanese Institute of Electronics, Information and Communication Engineers, Tokyo, 10/06/1988. 9 p.

[Boitet 90a] Christian Boitet, The evolution of ideas in classical MT: B. Vauquois' contribution to the theory and practice of MT (1960-1985), Proceedings ROCLING III (21-23/09/1990), Hsinchu, Taiwan. pp. 15-36.

[Boitet 90b] Christian Boitet, *Software and lingware engineering in recent (1980-1990) classical MT: Ariane-G5 and BV/areo/F-E*, Proceedings ROCLING III (21-23/09/1990) Hsinchu Taiwan. pp. 37-60.

[Boitet 90c] Christian Boitet, *Towards personal MT: general design, dialogue structure, potential role of speech, text encoding*, Proceedings ROCLING III (21-23/09/1990) Hsinchu Taiwan. pp. 61-70.

[Boitet 90d] Christian Boitet, *Multilingual Machine Translation does not have to be saved by Interlingua*, MMT'90, Tokyo, 5-6/11/1990, Panel "Can Interlingua be the savior of the multilingual machine translation world?". 2 p.

[Boitet 91a] Christian Boitet, *TAO & IA: Un système de traduction automatique doit-il et peut-il comprendre?*, acte de la convention IA-91, Paris, Hermès, 15-17/01/1991. 13 p.

[Boitet 91b] Christian Boitet, Quelle automatisation de la traduction peut-on souhaiter et realiser sur des stations de travail individuelles?, Colloque de Mons, réseau LTT, ANPELF-UREF, 25-27/04/1991. 12 p.

[Boitet 91c] Christian Boitet, *Twelve Problems for Machine Translation*, International Conference on Current Issues in Computational Linguistics, USM, Penang, 12-14/07/1991. 11 p.

[Boitet 93] Boitet C. & Blanchon H., *Dialogue-based machine translation for monolingual authors and the LIDIA project*. In H. Nomura, editor, Proceedings of the 1993 Natural Language Processing Rim Symposium, Fukuoka, December 1993. Kyushu Institute of Technology. pp. 208-222.

[Boitet 95a] Christian Boitet, *Factors for success (and failure) in Machine Translation – some lessons of the first 50 years of R&D*, Proceedings MT-SUMMIT V, Luxemburg, European Community, 11-13/07/1995. 18 p.

[Boitet 95b] Christian Boitet & Herve Blanchon, *Multilingual Dialogue-based MT for monolingual authors : the LIDIA project and a first mockup,* in Machine Translation, vol.9(2), 1995. pp.99-132.

[Boitet 97] Boitet Ch., GETA's MT Methodology and its Current Development towards Personal Networking Communication and Speech Translation in the Context of the UNL and C-STAR Projects, Proceedings PACLING '97 2-5/09/1997, Meisei University, Ohme, Tokyo, Japan. pp. 23-57.

[Boitet 99a] Boitet Ch., A research perspective on how to democratize machine translation and translation aids aiming at high quality final output, Proceedings MT Summit VII (1999), Singapore, 13-17/9/99. pp. 14-21.

[Boitet 99b] Boitet Ch., *Dialogue-Based MT and Self-explaining Documents as an Alternative to MAHT and MT of Controlled Language*, Machine Translation Review No. 10 October 1999, British Computer Society. pp. 6-15.

[Boitet 00] Boitet Ch., Blanchon, H, Guilbaud J.-P.(2000). *A way to integrate context processing in the MT component of spoken, task-oriented translation systems.* Proc. MSC2000. Kyoto, Japan. October 11-13, 2000. pp. 83-87.

[Boitet 01] Boitet Ch., *Four technical and organizational keys for handling more languages and improving quality (on demand) in MT*, MT-SUMMIT VIII (2001), Proceedings of the Workshop "Towards a Road Map for MT", 18/09/2001. pp.14-21.

[Boitet 02a] Boitet Ch., *A rationale for using UNL as an Interlingua and more in various domains.*, Proceedings "First International Workshop on UNL, other Interlinguas and their Applications", LREC2002, Las Palmas, Spain, 27/5-2/6/2002. pp. 23-26.

[Boitet 02b] Boitet Ch. & Tsai W.-J., La coédition langue _ UNL pour partager la révision entre les langues d'un document multilingue: un concept unificateur, Proceedings TALN2002, 24-27/06/2002, Nancy, France. pp. 275-286.

[Boitet 02c] Boitet Ch. & Tsai W.-J., *Coedition to share text revision across languages and improve MT a posteriori*, Proc. Post-Conference Workshops "Machine Translation in Asia", COLING2002, 1/9/2002, Taipei Taiwan. pp 9-19.

[Boitet 02d] Boitet Ch., *Proposals for solving some problems in UNL encoding*, International Conference on Universal Knowledge and Language (ICUKL2002), Goa, India, 25-29 November 2002. (slides)

[Boguslavsky 00] Boguslavsky I., Frid N., Iomdin L., Kreidlin L., Sagalova I. & Sizov V., *Creating a Universal Networking Language Module within an Advanced NLP System* , Proc. COLING 2000, Saarbrücken, Germany 31/07-04/08. pp.76-82.

[Boguslavsky 01a] Boguslavsky I., *Guidelines for writing UNL expressions*, FB2004 (ref FB2004 web site http://piramides.dia.fi.upm.es/fb2004/explorer.htm), June 2001. 17 p.

[Boguslavsky 01b] Boguslavsky I., *Additions to the Guidelines for writing UNL expressions*, FB2004 (ref FB2004 web site http://piramides.dia.fi.upm.es/fb2004/explorer.htm), August 2001. 11 p.

[Boguslavsky 02a] Bobuslavsky I., *Some Lexical Issues of UNL*, Proc. Workshop "First International Workshop on UNL, other Interlinguas and their Applications", LREC2002, Las Palmas, Spain, 27/5-2/6/2002. pp. 19-22.

[Boguslavsky 02b] Boguslavsky I., *Encoding UNL Expressions: Some Problems and Proposals*, International Conference on Universal Knowledge and Language (ICUKL2002), Goa, India, 25-29 November 2002.

[Bouayad-Agha 02] Bouayad-Agha N., Power R. Scott D. & Belz Anja, *PILLS: Multilingual Generation of Medical Information Documents with Overlapping Content*, Proc. LREC 2002, Las Palmas, Spain. pp. 2111-2114.

[Bourbeau 90] Bourbeau L., Carcagno D., Goldberg E., Kittredge R. & Polguère A.,,*Bilingual Generation of Weather Forecasts in an Operational Environment*, Proceedings of COLING-90, Helsinki, Finland. pp. 318-320.

[Brun 00] Brun C., Dymetman M. & Lux V., *Document Structure and Multilingual Authoring*, Proc. INLG-2000, Mitzpe Ramon, Israel. pp. 24-31.

[Cardeñosa 01a] Cardeñosa J., Iraola L., Tovar E., *Workplan of FB2004: a showcase of UNL deployment.* FB2004 project internal document, (ref FB2004 web site http://piramides.dia.fi.upm.es/fb2004/explorer.htm), 24/04/2001. 19 p.

[Cardeñosa 01b] Cardeñosa J., Iraola L., Tovar E., *Procedure of FB2004: a showcase of UNL deployment.* FB2004 project internal document, (ref FB2004 web site http://piramides.dia.fi.upm.es/fb2004/explorer.htm), 31/08/2001. 17 p.

[Chantriaux 03] Chantriaux J., De l'étude du formalisme linguistique des grammaires statiques à l'édition de correspondance chaîne-arbre, rapport de TER de maîtrise d'informatique, UJF, 19 mai 2003. 20 p.

[Chappuy 83] Chappuy S., Formalisation de la description des niveaux d'interprétation des langues naturelles. Etude menée en vue de l'analyse et de la

génération au moyen de transducteurs, Thèse de 3 cycle, INPG, Grenoble - 02/07/1983. 213 p.

[Chevreau 00] Chevreau K. & Coch J., *Génération Multilingue de Bulletins Météorologiques: le Logiciel MultiMeteo*, Procs 2eme Colloque Francophone de Génération Automatique de Textes (GAT-99), Grenoble, France.

[Choudhary 01] Choudhary Bh. & Bhattacharyya P., *Text Clustering Using Universal Networking Language*, First International UNL Open Conference, Suzhou, China, 26-29/11/2001. 7 p.

[Coch 01] Coch J. & Chevreau K., *Interactive Multilingual Generation*. Proc. CICLing-2001 (Computational Linguistics and Intelligent Text Processing), Mexico, February 2001, Springer, A. Gelbukh ed.. pp. 239-250.

[Cole 96] Cole R. (Editor in Chief), *Survey of the State of the Art in Human Language Technology*, Cambridge University Press ISBN 0-521-59277-1. 533 p.

[Czuba 98] Czuba K., Mitamura T., & Nyberg E., *Can Practical Interlinguas Be Used for Difficult Analysis Problems*, Proceedings AMTA-98 (Association for Machine Translation in Americas) Workshop on Interlinguas. 27/10/1998, Langhorne, Pennsylvania, USA. 9 p.

[Dave 02] Dave Sh., Parikh J. & Bhattacharyya P., *Interlingua-Based English-Hindi Machine Translation and Language Diverhence*, International Conference on Universal Knowledge and Language (ICUKL2002), Goa, India, 25-29 November 2002. 59 p.

[van Deemter 00] van Deemter K. & Power R., *Authoring Multimedia Documents using WYSIWYM Editing*, Proc. COLING-2000, Saarbruecken, Germany. pp. 222-228.

[van Deemter 98] van Deemter K. & Power R., *Coreference in Knowledge Editing*, Proceeding COLING 98, workshop on the Computational Treatment of Nominals, Montreal, Canada. pp. 56-60.

[Doi 92] Shinichi D. & Kazunori M., *Translation Ambiguity Resolution Based on Text Corpora of Source and Target Languages*. Proceedings COLING 92, 23-28/08/1992, Nantes, France. pp. 525-531.

[Ducrot 82] Ducrot J.-M., *TITUS IV*. In "Information research in Europe. Proc. of the EURIM 5 conf. (Versailles)", P. J. Taylor, ed., ASLIB, London.

[Ducrot 88] Ducrot J.-M., *Le système TITUS IV*. In “Traduction Assistée par Ordinateur. Actes du séminaire international sur la TAO et dossiers complémentaires”, A. Abbou, ed., Observatoire des Industries de la Langue (OFIL), Paris, mars 1988. pp. 55-71.

[Dymetman 00] Dymetman M., Lux V. & Ranta A., *XML and Multilingual Document Authoring: Convergent Trends*, Proceedings COLING-2000, Saarbrucken, Germany. pp. 243-249.

[Dymetman 02] Dymetman M., *Document Content Authoring and Hybrid Knowledge Bases*, Proceedings KRDB-02, Toulouse, France. 13 p.

[Goodman 89] Goodman K. & S. Nirenburg (eds.) *KBMT-89 Project Report*. Carnegie Mellon University. Center for Machine Translation. 286 p.

[Goodman 92] Goodman K. & S. Nirenburg (eds.) *KBMT-89: a Case Study in Knowledge-Based Machine Translation*. San Mateo, California, Morgan Kaufmann, 331 p.

[Grasson 96] Katty Grasson, *Grammaire statique et typage textuel*. Rapport de stage de DEA d'informatique: système et communication, ENSIMAG, INPG, Grenoble, 19/06/1996. 92 p.

[Guzman de Rojas 88] Guzman de Rojas I., *ATAMIRI – interlingua MT using Aymara language*. in Maxwell D., Schubert K., Witkam A.P., editors, New Directions in Machine Translation. Foris Publishers. pp. 123-130.

[Hajlaoui 03] Hajlaoui N. & Boitet Ch., A "pivot" XML-based architecture for multilingual, multiversion documents : parallel monolingual documents aligned through a central correspondence descriptor and possible use of UNL, Convergence 03, International Conference of the Convergence of Knowledge, Culture, Language and Information Technology, 2nd-6th December, 2003, Alexandria Library, Alexandria, Egypt, 8 p.

[Hartley 95] Hartley A. & Paris C., *Supporting Multilingual Document Production: Machine Translation or Multilingual Generation?*, Working notes of the Multilingual Text Generation workshop, Proceedings International Joint Conference in Artificial Intelligence (IJCAI 95), Montreal, Canada. pp. 34-41.

[Hartley 01] Hartley A., Scott D., Bateman J.& Dochev D., *AGILE - A System for Multilingual Generation of Technical Instructions*, Proceedings MT-Summit VIII(2001), Santiago de Compostella, Spain. pp. 145-150.

[Hasan 01] Sirin Hasan & Mohammed Khair Odeh, *Distributed UNL Proxy*, First International UNL Open Conference, Suzhou, China, 26-29/11/2001. 3 p.

[Helme 02] Helme S., *Coéd – Système de coédition utilisant UNL.* Rapport de stage d'été de DESS Double Compétence Informatique et Sciences Sociales, Université Pierre Mendès-France, juillet-septembre 2002. 45 p.

[Hong 99] Hong M.-P.& Streiter P., *Overcoming the Language Barriers in the Web: The UNL-Approach*, 11-e Jahrestagung der Gesellschaft für linguistische Datenverarbeitung (GLDV'99), 1999, Frankfurt am Main. 10 p.

[Hovy 01] Hovy E., Ide N., Frederking R., Mariani J. & Zampolli A. (editors) , Linguistica Computazionale, Volume XIV-XV*, "Multilingual Information Management: Current Levels and Future Abilities",* Publisher: Instuti Editoriali e Poligrafici Internazionali, Pisa, Italy, 2001. ISSN 0392-6907. ref. http://www-2.cs.cmu.edu/~ref/mlim/.

[Hung 04] Hung V.-Tr., Réutilisation de traducteurs gratuits pour développer des systèmes multilingues, TALN RECITAL 2004, avril 2004, Fès, Maroc. 6 p.

[Hutchins 88] Hutchins J., *Recent developments in machine translation*, New Directions in Machine Translation, conference proceedings, Budapest 18-19 August, 1988. pp. 7-64.

[Hutchins 93] Hutchins J., *Latest Developments in Machine Translation technology: Beginning a New Era in MT research*, Proceedings MT Summit IV.: International cooperation for global communication, July 20-22, 1993, Kobe, Japan. pp. 11-34.

[Hutchins 95] Hutchins J., *Machine Translation: a brief history*. From "Concise history of the language science: from the Sumerians to the cognitivists. Edited by E.F.K. Koerner and R.E. Asher, Oxford, Pergamon Press 1995. pp. 431-445.

[Hutchins 99] Hutchins J., *The development and use of machine translation systems and computer-based translation tools*. Processing International Conference on Machine Translation & Computer Language Information, 26-28 June 1999, Beijing, China. pp. 1-16.

[Hutchins 01] Hutchins J., *Towards a new vision for MT*. Introductory speech at the 'MT Summit VIII' conference, 18-22 September 2001, Santiago de Compostela, Galicia, Spain. 6 p.

[Hutchins 02] Hutchins J., *Machine Translation and Translation Aides: systems, problems, uses, prospects*. Power points slides for Università di Bologna, SSLMIT, Forlí, December 2002.

[Iordanskaja 92] Iordanskaja L., Kim M, Kittredge R, Lavoie B. & Polguere A., *Generation of Extended Bilingual Statistical Reports*, Proceedings COLING-92, Nantes, France. pp. 1019-1023.

[Jitkue 01] Jitkue P., Participation au projet SWIIVRE-UNL et première version d'un environnement Web de déconversion multilingue et d'éditeur UNL de base. Rapport de stage de Maîtrise d'informatique, Université Joseph Fourier, septembre 2001. 23 p.

[Kahane 99] Kahane S. Mel' _uk Igor, Synthèse des phrases à extraction en français contemporain (Du graphe sémantique à l'arbre de dépendance), T.A.L., 40:2. pp. 25-85.

[Kahane 00] Kahane S., *Des grammaires pour définir une correspondance*, Proceedings TALN 2000, Lausanne, 16-18 octobre 2000. pp. 197-206.

[Kahane 01] Kahane S., *What is a Natural Language and How to Describe It? Meaning Text Approach in Contrast with Generative Approaches*, in Proceedings of second International Conference of Computational Linguistics and Intelligent Text Processing (CICLing), Mexico, 2001. pp. 1-17.

[Kittredge 98] Kittredge R. & Lavoie B., *METEOCOGENT: A Knowledge-based Tool for Generating Weather Forecast Texts*, American Meteorological Society Conference (AMS-98), Phoenix, Arizona. 5 p.

[Kruijff 00] Kruijff G.-J.,Teich E., Bateman J. & Kruijff-Korbayov I., *Multilinguality in a Text Generation System for Three Slavic Languages*, Proceedings COLING 2000, Saarbrucken, Germany. pp. 474-480.

[Laubsch 84] Laubsch J., Roesner D.; Hanakata K.; Lesnlewski A., *Language Generation from Conceptual Structure: Synthesis of German in a Japanese/ German MT Project*, Proceedings COLING 84. pp. 491-494.

[Lavie 01a] Lavie A., Levin L, Schultz T., Langley C., Han B., Tribble A., Gates D., Wallace D., Peterson K. (Carnegie Mellon University, USA), *Domain Portability in Speech-to-Speech Translation*. Proc. HLT 2001 (Human Language Technology Conference) , San Diego, California, USA, 18-21/03/2001. 5 p.

[Lavie 01b] Lavie A., Langley C., Waibel A. (Carnegie Mellon University, USA), Pianesi F., Lazzari G., Coletti P. (ITC-irst, Italy), Taddei L., Balducci F. (AETHRA, Italy), *Architecture and Design Considerations in NESPOLE!: a Speech Translation System for E-commerce Applications*. Proc.HLT 2001 (Human Language Technology Conference), San Diego, USA. 18-21/03/2001. 4 p.

[Lepage 86] Lepage Y., *A Language for Transcriptions*. Proc. COLING-86, Bonn, Germany, August 1986. pp. 402-404.

[Lepage 88] Lepage Y. & Zaharin Y., String-Tree Correspondences, Identification and linguistic description, PTMK , USM, April 1988. 9 p.

[Lepage 89] Lepage Y., *Un système de grammaires correspondancielles d'identification.* Nouvelle thèse, UJF, Grenoble, 14/06/89. 184 p.

[Lepage 91] Lepage Y. & Zaharin Y., *Identification: a unification-like operation with variables instantiating to forests*, Proceedings of the International Conference on Current Issues in Computational Linguistics, Penang, June 1991. 14 p.

[Levin 98] Levin L., Gates D., Lavie A. and Waibel A., *An Interlingua Based on Domain Actions for Machine Translation of Task-Oriented Dialogues*. Proceedings ICSLP-98, Sydney, Australia, November 1998. 4 p.

[Levin 00a] Levin, L., Gates D., Wallace D., Bartlog B., Lavie A., Watanabe T., M. Woszczyna, and A.F. Llitjos, *Lessons learned from a Task-Based Evaluation of Speech-to-Speech Machine Translation*. Proceedings LREC 2000, Athens, Greece, July 2000. 4p.

[Levin 00b] Levin, L., Gates D., Wallace D., Lavie A., Pianesi F., Watanabe T. & Woszczyna M., *Evaluation of a Practical Interlingua for Task-Oriented Dialogue*. Proceedings SIG-IL Workshop at the NAACL 2000, Seattle, USA, July 2002. pp. 18-23.

[Levin 02] Levin, L., Gates D., Wallace D., Peterson K., Lavie A., Pianesi F., Pianta E., Cattoni R. & Mana N., *Balancing Expressiveness and Simplicity in an Interlingua for Task-Based Dialogue.* Proceedings Speech-to-Speech Translation Workshop at the 40th Annual Meeting of the Association of Computational Linguistics (ACL-02), Philadelphia, PA, July 2002. 8 p.

[Levin 03] Levin L., Gates D., Wallace D., Peterson K., Pianta E., Mana N., *The NESPOLE ! Interchange Format*, Final report, 24/02/2003. 66 p.

[Linden 95] Linden K. V. & Scott D., *Raising the Interlingual Ceiling with Multilingual Text Generation*, Proceedings Multilingual Natural Language Generation workshop (IJCAI-95), Montreal, Canada. pp. 95-109.

[Lonsdale 94] Lonsdale D. W., Franz A. & Leavitt J., *Large-scale Machine Translation: An Interlingua Approach*, Proceedings of the Seventh International

Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, May 31-June 3, 1994, Austin, Texas. ACM, 1994. pp. 515-523.

[Mangeot-Lerebours 01] Mangeot-Lerebours M., Environnement centralisés et distribués pour lexicographes et lexicologues en contexte multilingue. Nouvelle thèse, UJF, Grenoble, 27/09/01. 279 p.

[Martins 99] Martins R., Dos Problemas da ambiguidade semântica em um modelo de traduçao automatica baseado em interligua : Apontamentos do projeto UNL-Brasil, Computational Processing of Portuguese Symposium, 30/04/1999-1/5/1999, São Paulo, Brazil. 6 p.

[Martins 00] Martins R.T., Rino, L.H.M., Nunes, M.G.V., Montilha G., Oliveira Jr. O.N., *An interlingua aiming at communication on the Web: How language-independent can it be?* Workshop on Applied Interlinguas: Practical Applications of Interlingual Approaches to NLP (Pre-Conference Workshop in conjunction with ANLP-NAACL2000). April 30, 2000. Seattle, Washington, USA. pp. 24-30.

[Martins 02] Martins, R.T., Rino, L.H.M., Nunes. M.G.V., Oliveira Jr. O.N., *The UNL distinctive features: evidences through a NL-UNL encoding task.* The First International Workshop "on UNL, other Interlinguas and their Applications." Proc. LREC 2002. Las Palmas, Canary Islands, Spain. 29-31 May 2002. pp. 08-13.

[Mel'_uk 65] _olkovskij A. & Mel'_uk I., O vozmo_nom metode I instrumentax semanti_eskogo sinteza [On a possible method and instruments for the semantic synthesis (of text)], Nau_no-texni_eska informacija [Scientific and Technological Information], 6. pp. 23-28.

[Menezes 01] Menezes A. & Richardson S. D., *A best-first alignment algorithm for automatic extraction of transfer mappings from bilingual corpora*. In Proceedings Workshop on Data-driven Machine Translation at 39th Annual Meeting of the Association for Computational Linguistics, Toulouse, France, 2001. pp. 39-46.

[Mitamura 93] Mitamura T., Nyberg E. H., Carbonell J. G., *Automated Corpus Analysis and the Acquisition of Large, Multi-Lingual Knowledge Bases for MT.* Proc. 5th International Conference on Theoretical and Methodological Issues in Machine Translation, Kyoto, Japan, July 14-16, 1993, 17 p.

[Miura 92] Miura M., Hirata M., & Hoshino N., *Learning Mechanism in Machine Translation System "PIVOT"*, Proceedings COLING-92, 23-28 August, Nantes France. pp. 693-699.

[Morneau 92] Morneau R., *On the unsuitability of "logical languages" for use as interlingua in machine translation*, from the 14th edition of Journal of Planned Languages (1992). Also from http://www.invisiblelighthouse.com/langlab/mtil.html. 3 p.

[Muraki 87] Kawunori M., *PIVOT: Two-Phase Machine Translation System,* Proceedings MT-SUMMIT (I), Hakone, Japan. pp. 113-115.

[Nicholas 96] Nicholas N., *Lojban as a Machine Translation Interlanguage in the Pacific*, Proceedings of the 4th Pacific Rim International Conference on the Artificial Intelligence: Workshop on "Future Issue for Multilingual Text Processing", Cairns, Australia, 27/08/1996. pp. 31-39.

[Nirenburg 89] Nirenburg S., *KBMT-89 – A Knowledge-Based MT Project at Carnegie Mellon University*, Proceedings MT-SUMMIT II, 16-18 August 1989, Munich, Germany. pp. 141-147.

[Nirenburg 90] Nirenburg S., *Lexical and conceptual structure for knowledge-based machine translation*, Proceedings ROCLING-III (21-23/09/1990) Hsinchu Taiwan. pp. 103-130

[Nirenburg 98] Nirenburg S., Raskin V., *Universal Grammar and Lexis for Quick Ramp-Up of MT Systems*, Proceedings COLING 98 (10-14/08/1998) Montreal, Canada. pp. 975-981.

[Nunes 01] des Graças M., Nunes V., Martins R. T., Rino L., Oliveira Jr. O., *The decoding system for Brazilian Portuguese using the Universal Networking Language (UNL)*, First International UNL Open Conference, Suzhou, China, 26-29/11/2001. 14 p.

[Nyberg 92] Nyberg E. H. & Mitamura T., *The KANT system: Fast, Accurate, High-Quality Translation in Practical Domains.* Proc. COLING-92, Nantes, 23-28 July 92, Ch. Boitet, ed., ACL, vol. 3/4. pp. 1069-1073.

[Nyberg 97] Nyberg E. H. & Mitamura T & Carbonell J., *The KANT Machine Translation System: From R&D to Initial Deployment,* Presentation at LISA Workshop on Integrating Advanced Translation Technology, Washington, D.C., June 3-4. 7 p.

[Odijk 89] Odijk J., *The Organization of the Rosetta Grammars*, Proceedings of 4th EACL (European Association for Computational Linguistics) conference, 10-12/April, 1989, UMIST, Manchester, UK. pp. 80-86.

[Okumura 91] Okumura A., Muraki K., and Akamine S., *Multi-lingual sentence generation from the PIVOT interlingua*., Proceedings MT-SUMMIT 91(III), Washington, USA. pp. 67-71.

[Okumura 94] Okumura A. & Muraki K., *Symmetric pattern matching analysis for English Coordinate Structures*. Proceedings 4th Conference Applied NLP, 1994. pp. 41-46.

[Onyshkevych 91] Onyshkevych B., Nirenburg S.: *Lexicon, Ontology, and Text Meaning*, Lexical Semantics and Knowledge Representation, Proceedings First SIGLEX Workshop, Berkeley, CA, USA, June 17, 1991. pp. 238-249.

[Papegaaij 86] Papegaaij B.C., Sadler V. & Witkam A.P.M, *Experiments with an MT-Directed Lexical Knowledge Bank.*, Proceedings COLING 1986, Bonn, Germany. pp. 432-434.

[Paris 95a] Paris C., Linden K. V., Fischer M., Hartley A., Pemberton L., Power R. & Scott D, *A Support Tool for Writing Multilingual Instructions*, Proceedings PAIJCAI 95, Montreal, Canada. pp. 1398-1404.

[Paris 95b] Paris C. & Scott D., *DRAFTER: Support for the Production of Multilingual Instructions*, Proceedings 2nd Language Engineering Conference, October 1995, London, UK. 8 p.

[Planas 98] Planas E., *TELA – Structure et algorithmes pour la traduction fondée sur la mémoire*. Nouvelle thèse, UJF, Grenoble, 07/07/98. 376 p.

[Power 98a] Power R. & Scott D., *Multilingual Authoring using Feedback Texts*, Proceedings COLING/ACL-98, Montreal, Canada. pp.1053-1059.

[Power 98b] Power R., Scott D. & Evans R., *What You See Is What You Meant: Direct Knowledge Editing with Natural Language Feedback*, Proceedings 13th European Conference on Artificial Intelligence (ECAI-98), Brighton, UK. 10 p.

[Rösner 86] Rösner D., When Mariko Talks to Siegfried – Experiences from a Japanese/German Machine Translation Project., Proceedings COLING 1986, Bonn, Germany. pp 652-654.

[Rösner 94] Rösner D. & Stede M., *Generating Multilingual Documents from a Knowledge Base: The TECHDOC Project*, Proceedings COLING-94,Kyoto, Japan. pp. 339-343.

[Schubert 86] Schubert K., Linguistic And Extra-Linguistic Knowledge. A catalogue of language-related and their computational application in machine translation. in Computers and Translation. Vol. I(3). pp. 125-152.

[Schubert 88a] Schubert K., *The architecture of DLT – interlingual or double direct?*, Proceedings New Directions in Machine Translation, Budapest, 18-19 August, 1988. pp. 131-143.

[Schubert 88b] Schubert K., *Implicitness as a Guiding Principle in Machine Translation.* Proc. COLING-88, Budapest, 22–27 Aug. 1988. pp. 599-601.

[Scott 99] Scott D., The Multilingual Generation Game: Authoring Fluent Texts in Unfamiliar Languages, Proceedings IJCAI 1999, Stockholm, Sweden. 5 p.

[Sheremetyeva 96] Sheremetyeva S., Nirenburg S.& Nirenburg I., *Generating Patent Claims from Interactive Input*, Proceedings 8th International Workshop on Natural Language Generation, Herstmonceux, UK.

[Shieber 90] Shieber S., Schabes Y., *Synchronous Tree-Adjoining Grammars*, Proceedings of COLING 1990, Helsinki 1990. pp. 253-258.

[Sérasset 94a] Sérasset G., SUBLIM : un système universel de bases lexicales multilingues et NADIA : sa specialisation aux bases lexicales interlingues par acceptions , Thèse préparée au sein du laboratoire GETA-IMAG, Grenoble - 08/12/1994. 194 p.

[Sérasset 94b] Sérasset G., *Interlingual Lexical Organisation for Multilingual Lexical Databases in NADIA*, Proceedings COLING 94, 5-9 August 1994, Kyoto, Japan. pp. 278-282.

[Sérasset 99] Sérasset G. & Boitet Ch., UNL-French deconversion as transfer & generation from an interlingua with possible quality enhancement through offline human interaction, Proc. MT Summit VII (1999), 13-17 September 1999, Singapore. pp. 220-228.

[Sérasset 00] Sérasset G. & Boitet Ch., On UNL as the future ”html of the linguistic content” & the reuse of existing NLP components in UNL-related applications with the example of a UNL-French deconverter, Proceedings COLING 2000, 31/07-04/08, Saarbrücken, Germany. pp. 768-774.

[Song 02] Song X., *Construire la base de données des corpus UNL.* Rapport de stage de Maîtrise d'informatique, Université Joseph Fourier, mai 2002. 16 p.

[Sornlertlamvanich 00] Sornlertlamvanich V., Potipiti T. & Charoenporn Th., *Thai Lexical Semantic Annotation by UW*, WAINS7, Bangkok, Thailand, December 2000. 6 p.

[Sornlertlamvanich 01] Sornlertlamvanich V., Potipiti T. & Charoenporn Th., *UNL Document Summarization*, Proceedings of the First International Workshop on MultiMedia Annotation (MMA2001), Tokyo, Japan, January 2001. 5 p.

[Streiff 85] Streiff A.A., *New Developments in TITUS 4*, from “Tools for the trade, translating and the computer” Lawson, V. (ed.) 1985. pp. 185-192.

[Sugiyono 01] Sugiyono M. Hum., *Machine Translation and Man-Machine Interface Development in Indonesia*, International symposium on language in cyberspace, 26-27 September 2001, Seoul South Korea. 5 p.

[Tait 97] Tait J., Sanderson H., Hellwig P., Ellman J., Tsahageas P. & San Jos A. M. M., *Practical Considerations in Building a Multilingual Authoring System for Business Letters*, Proceedings Workshop on Commercial Applications of NLP (ACL/EACL-97), Madrid, Spain.

[Tait 99] Tait J. & Ellman J., *MABLe: a Multilingual Authoring Tool for Business Letters*, Proceedings of the 21st International Conference on Translating and the Computer, ICTC99, London, UK.

[Takeda 90] Takeda K., *Pattern-Based Machine Translation*, Proceedings of COLING 1990, Helsinki. pp. 1155-1158.

[Tanaka 89] Tanaka H., Ishizaki Sh., Uehara A. & Uchida H., *Research and development of cooperation project on a machine translation system for Japan and its neighboring countries*. Proceedings MT-SUMMIT 89 (II), 16-18/08/1989, Munich, Germany. pp. 147-152.

[Tang 94] Tang E. K., Natural Language Analysis in Machine Translation (MT) Based on the String-Tree Correspondence Grammar (STCG), Ph. D. Thesis, UTMK, Universiti Sains Malaysia 1994. 253 p.

[Tang 95] Tang E. K. & Zaharin Y., *Handling Crossed Dependencies with the STCG*, Proc. Natural Language Processing Pacific Rim Symposium, Sofitel Ambassador Hotel, Seoul, 4-6th December 1995. 7 p.

[Tomokiyo 01] Tomokiyo M., Al Assimi A. & Boitet Ch., Multilingual documents management by using Universal Networking Language UNL and an Alignment Gestion Tool OGA , Proc. PACLING2001, Fukuoka, Japan. 6 p.

[Tsai 01] Tsai W.-J., SWIIVRE - a Web Site for the Initiation, Information, Validation, Research and Experimentation on UNL (Universal Networking Language), First International UNL Open Conference, Suzhou, China, 26-29/11/2001. 9 p.

[Tsai 02] Tsai W.-J., *A Platform for Experimenting UNL (Universal Networking Language)*, Workshop "First International Workshop on UNL, other Interlinguas and their Applications", Proceedings LREC2002, Las Palmas, Spain, 27/5-2/6/2002. pp. 27-32.

[Tsujii 86] Tsujii J., *Future Direction of Machine Translation*, Proceedings COLING 1986, Bonn, Germany. pp. 655-668.

[Tsujii 88] Tsujii J., *What is a cross-linguistically valid interpretation of discourse?*, New Directions in Machine Translation, conference proceedings, Budapest 18-19 August, 1988. pp. 7-64.

[Tsujii 90] Tsujii J., *Why do we need man-machine interaction in MT?* , Proceedings ROCLING-III (21-23/09/1990), Hsinchu, Taiwan. pp. 131-138.

[Uchida 80] Uchida H. & Sugiyama K., A Machine Translation System from Japanese into English Based on Conceptual Structure. Proceedings COLING 80, Tokyo. pp. 455-462.

[Uchida 89] Uchida H., *ATLAS-II: A machine translation system using conceptual structure as an interlingua*. Proceedings MT-SUMMIT 89 (II), 16-18 August 1989, Munich, Germany. pp. 153-157.

[Uchida 93] Uchida H., Zhu M.-Y., *Interlingua for Multilingual Machine Translation*, Proceedings of MT-SUMMIT IV, Kobe Japan. pp. 157-169.

[Uchida 01] Uchida H., *The Universal Networking Language beyond Machine Translation*, International symposium on language in cyberspace, 26-27 September 2001, Seoul South Korea. 14 p.

[UNL DeConverter 97] DeConverter Specifications. Version 1.0 (Tech. Rep. UNL-TR1997-010). UNU/IAS/UNL Center Tokyo, Japan. 25 p.

[Vasconcellos 88] Vasconcellos M. & Leon M., *SPANAM and ENGSPAN: Machine Translation at the Pan American Health Organisation*. In "Machine Translation Systems" edited by Slocum, Cambridge University Press. pp. 187-236.

[Vauquois 68] Vauquois B., *A survey of formal grammars and algorithms for recognition and transformation in machine translation*, Proceedings IFIP congress-68, Edinburgh, Scotland, August, 1968. pp. 254-260.

[Vauquois 69] Vauquois B., Veillon G., Nedobejkine N. & Bourguignon C., *Une notation des textes hors des contraintes morphologiques et syntaxiques de l'expression*, Proceedings COLING-69, Stockholm, Sweden. 27 p.

[Vauquoi 85a] Vauquois B. & Chappuy S., *Static grammars: a formalism for the description of linguistic models*. Proc. TMI-85 (Conf. on theoretical and methodological issues in the Machine Translation of natural languages), Aug. 1985. pp. 298-322.

[Vauquois 85b] Vauquois B. & Boitet Ch., *Automated Translation at Grenoble University*, Computational Linguistics, Volume 11, Number 1, January-March 1985. pp. 28-36.

[Watanabe 00] Watanabe H., Kurohashi S. & Aramaki E., *Finding Structural Correspondences from Bilingual Parsed Corpus for Corpus-based Translation*, Proceedings COLING-2000, August 2000. pp. 906-912.

[Witkam 88] Witkam T., *DLT - an industrial R&D project for multilingual machine translation*, Proceedings COLING-88, Budapast, Hungary. pp. 756- 759.

[Zaharin 86a] Zaharin Y., *Strategies and heuristics in the analysis of natural language in machine translation*. Ph. D. Thesis, USM, Penang (Research done at GETA, CNRS & UJF). 327 p.

[Zaharin 86b] Zaharin Y., Strategies and heuristics in the analysis of a natural language in Machine Translation (in the memory of Bernard Vauquois). Proc. COLING-86, Bonn, August, 1986. pp. 136-139.

[Zaharin 86c] Zaharin Y., *The Tree Correspondence Grammar: The Static Grammar Revisited*, document interne du GETA, mai 1986. 17 p.

[Zaharin 87] Zaharin Y., String-Tree Correspondence Grammar: a declarative grammar formalism for defining the correspondence between strings of terms and tree structures, Proceedings of the 3rd Conference of the European Chapter of the Association of Computational Linguistics, Copenhagen, April 1987. pp. 160-166.

[Zaharin 89] Zaharin Y., On Formalisms and Analysis, Generation and Synthesis in Machine Translation, Proceedings of the 4th Conference of the European Chapter of the Association of Computational Linguistics, Manchester, April 1989. pp. 319-326.

[Zajac 86a] Zajac R., Etude des possibilités d'interaction homme-machine dans un processusde traduction automatique. Nouvelle thèse, INPG, Grenoble, 17/07/86. 259 p.

[Zajac 86b] Zajac R., SCSL: a linguistic specification language for MT. Proceedings COLING-86, Bonn, August. 1986, pp. 393-398.

[Zajac 88] Zajac R., Interactive translation: a new approach, Proceedings COLING-88, Budapest, Hungary. pp. 785- 790.

[___Feng J.-W. 94] ____ "__________ (Ziran yuyan Jiqi Fanyi Xinlun)" _____ 1994_ ISBN:7-80006-744-0.

Signets

[UNL foundation] http://www.undl.org/

[UNL] http://www.unl.ias.unu.edu/

[FB2004] http://piramides.dia.fi.upm.es/fb2004/explorer.htm

[SWIIVRE-UNL] http://www-clips.imag.fr/geta/User/wang-ju.tsai/welcome.html

[WYSIWYM] http://www.itri.bton.ac.uk/projects/wysiwym

[ICONOCLAST] http://www.itri.bton.ac.uk/projects/iconoclast

[PILLS] http://www.itri.bton.ac.uk/projects/pills

[CLIME] http://www.itri.bton.ac.uk/projects/clime

[DRAFTER] http://www.itri.bton.ac.uk/projects/drafter

[Model Explainer] http://www.cogentex.com/research/modex/index.shtml

[MDA] http://www.xrce.xerox.com/competencies/content-analysis/dcm/

[Multimeteo] http://www.hltcentral.org/projects/detail.php?acronym=multimeteo

[MULTICOM] http://www-clips.imag.fr/multicom/

[C-STAR] http://www.c-star.org

[C-STAR II] http://www-clips.imag.fr/projets/cstar/IntroCstar.html

[PAPILLON] http://www.papillon-dictionary.org

[NESPOLE!] http://www.nespole.itc.it

[Interlingvo] http://ourworld.compuserve.com/homepages/profcon/e_dlt2.htm

[KANT] http://www.lti.cs.cmu.edu/Research/Kant/

[Horn 95] http://www.halcyon.com/horn/pages/o_am2.htm

[CICC] http://www.cicc.or.jp/english/kyoudou/mt.html

[Conceptual Graph (SOWA)] http://users.bestweb.net/~sowa/cg/

[EDR] http://www.jsa.co.jp/EDR/

[loglangs] http://www.geocities.com/Athens/Agora/7070/loglangs.htm

[UTL] http://www.xente.mundo-r.com/utl/

[Lojban] http://www.loglan.org

[artificial language] http://www.invisiblelighthouse.com/langlab/index.html

[MSDN] http://msdn.microsoft.com/downloads/samples/

Signets

[Mandarin Tools] http://www.mandarintools.com

[Chinese Computing] http://www.chinesecomputing.com

[Autotag CKIP _____] http://godel.iis.sinica.edu.tw/CKIP/ws/

[PILAF] http://clips.imag.fr/trilan/Pilaf/

[SILFIDE] http://silfide.imag.fr/

[FIPSTAG] http://www.latl.unige.ch

[Jasmine] http://www.cuhk.aoeit.org/Prog/basicchlnt.htm

[ICTCLAS] http://www.nlp.org.cn/project/project.php?proj_id=6

[MeCab] http://www.cl.aist-nara.ac.jp/~taku-ku/software/mecab/#install-windows

[ChaSen] http://chasen.aist-nara.ac.jp

[JUMAN] http://pine.kuee.kyoto-u.ac.jp/nl-resource/juman.html

[KAKASI] http://kakasi.namazu.org

[ANTLR] http://www.antlr.org

[Herein] http://www.european-heritage.net/sdx/herein/index.xsp

[la main à la pâte] http://www.inrp.fr/lamap

[UNESCO] http://www.unesco.org

[Enhydra] http://www.enhydra.org

[Dicoweb] http://www-clips.imag.fr/geta/services/dicoweb/dicoweb.html

Annexe A□ Spécifications d'UNL

Les spécifications suivantes sont tirées de la version 3 édition 1 datées du 20/05/2002.

Syntaxe d'un document UNL en expression BNF (UNL-html.1)

<UNL document> ::= "[D:" <dinf> "]" { "[P:" <number> "]" { "[S:" <number> "]" <sentence> "[/S]" }... "[/P]" }... "[/D]"

<dinf> ::= <document name> "," <owner name> ["," <document id> "," <date> "," <mail address>]

<document name> ::= "dn=" <character string>

<owner name> ::= "on=" <character string>

<document id> ::= "did=" <character string> /* defined by system */

<date> ::= "dt=" <character string> /* defined by system */

<mail address> ::= "mid=" <character string> /* defined by system */

<sentence> ::= "{org:" <l-tag> ["=" <code>] "}" <source sentence> "{/org}" "{unl" [":" <uinf>] "}" <UNL expression> "{/unl}" "{" <l-tag> ["=" <code>] [":" <sinf> "]" <generated sentence> "{/" <l-tag> "}"

/* necessary information about one sentence */

<l-tag> ::= "ab" | "cn" | "de" | "el" | "es" | "fr" | "id" | "hd" | "it" | "jp" | "lv" | "mg" | "pg" | "ru" | "sh" | "th" /* language flag */

<code> ::= <character code name>

<character code name>::= <character string>

<source sentence> ::= <character string>

<generated sentence>::= <character string>

<uinf> ::= <system name> "," <post editor name> "," reliability> ["," <date> "," <mail address>]

<sinf> ::= <system name> "," <post editor name> "," reliability> ["," <date> "," <mail address>]

<system name> ::= "sn=" <character string>

<post editor name> ::= "pn=" <character string>

<reliability> ::= "rel=" <digit>

<number> ::= <digit> /* sentence number */

The tags used in the above definition are the following.

[D:<dinf>]	indicates the beginning of a document and the

	necessary information about the document
[/D]	indicates the end of a document
[P:<number>]	indicates the beginning of a paragraph
[/P]	indicates the end of a paragraph
[S:<number>]	indicates the beginning of a sentence and the sentence number
[/S]	indicates the end of a sentence
{org:<l-tag>[=<code>]}	indicates the beginning of an original/source sentence, language and character code.
{/org}	indicates the end of an original sentence
{unl[:<uinf>]}	indicates the beginning of the UNL expressions of a sentence and necessary information.
{/unl}	indicates the End of the UNL expressions of a sentence

Syntaxe d'UW en EBNF (Extended BNF, BNF étendue)

<UW> ::= <Head Word> [<Constraint List>]

<Head Word> ::= <character>

<Constraint List> ::= "(" <Constraint> ["," <Constraint>]… ")"

<Constraint> ::= <Relation Label> { ">" | "<" } <UW> [<Constraint List>]
| <Relation Label> { ">" | "<" } <UW> [<Constraint List>]

[{ ">" | "<" } <UW> [<Constraint List>]]

<Relation Label> ::= "agt" | "and" | "aoj" | "obj" | "icl" | ..| "to" | "via"

<character> ::= "A" | ... | "Z" | "a" | ... | "z" | 0 | 1 | 2 | ... | 9 | "_" | " " | "#" |

"!" | "$" | "%" | "=" | "^" | "~" | "|" | "@" | "+" | "-" | "<" |

">" | "?"

Syntaxe des relations binaires en EBNF

<Binary Relation> ::= <Relation Label> [":"<Compound UW-ID>] "("

{{ <UW1> [":" <UW-ID1>]} | { ":" <Compound UW-ID1> }}[<Attribute List>] ","{{ <UW2> [":" <UW-ID2>]} | { ":" <Compound UW-ID2> }}[<Attribute List>] ")"

<Attribute List> ::= { "." <Attribute label> }

<Attribute Label> ::= "@entry" | "@may" | "@past" | ...| "@wish" | "@yet"

<UW-ID> ::= <alphanum><alphanum>

<alphanum> ::= "A"|"B"|"C"|…|"Y"|"Z"|"0"|"1"|….|"9"

<Compound UW-ID>::= "00"|"01"|"02"|….|"98"|"99"

/* 00 is used for representing the main scope, which can be omitted.*/

Liste des relations UNL

UNL Specifications version 3 Edition 1, December 2002

agt	agent	a thing in focus which initiates an action
and	conjunction	a conjunctive relation between concepts
aoj	thing with attribute	a thing which is in a state or has an attribute
bas	basis	a thing used as the basis (standard) for expressing a degree
ben	beneficiary	an indirectly related beneficiary or victim of an event or state
cag	co-agent	a thing not in focus which initiates an implicit event which is done in parallel
cao	co-thing with attribute	a thing not in focus, as in a state in parallel
cnt	content	an equivalent concept
cob	affected co-thing	a thing which is directly effected by an implicit event done in parallel or an implicit state in parallel
con	condition	a non-focused event or state which conditions a focused event or state
coo	co-occurrence	a co-occurrent event or state for a focused event or state
dur	duration	a period of time during which an event occurs or a state exists
fmt	range	a range between two things
frm	origin	an origin of a thing
gol	goal/final state	the final state of an object or the thing finally associated with the object of an event
ins	instrument	the instrument to carry out an event
man	manner	the way to carry out an event or characteristics of a state
met	method	the means to carry out an event
mod	modification	a thing which restricts a focused thing
nam	name	a name of a thing
obj	affected thing	a thing in focus which is directly effected by an event or state
opl	affected place	a place in focus where an event takes effect
or	disjunction	a disjunctive relation between two concepts
per	proportion, rate of distribution	a basis or unit of proportion, rate of distribution
plc	place	the place where an event occurs, or a state is true, or a thing exists
plf	initial place	the place where an event begins or a state becomes true

plt	final place	the place where an event ends or a state becomes false
pof	part-of	a concept of which a focused thing is a part
pos	possessor	the possessor of a thing
ptn	partner	an indispensable non-focused initiator of an action
pur	purpose or objective	the purpose or objective of an agent of an event or the purpose of a thing which exists
qua	quantity	quantity of a thing or unit
rsn	reason	a reason why an event or a state happens
scn	scene	a virtual world where an event occurs, or a state is true, or a thing exists
seq	sequence	a prior event or state of a focused event or state
src	source/initial state	the initial state of an object or thing initially associated with the object of an event
tim	time	the time an event occurs or a state is true
tmf	initial time	the time an event starts or a state becomes true
tmt	final time	the time an event ends or a state becomes false
to	destination	a destination of a thing
via	intermediate place or state	an intermediate place or state of an event

The following relation is used only in the UNL KB or UW definition.

icl	included	a concept of which a focused concept is a proper subset
iof	instance of	an instance of a class
equ	equal	an acronym of an original word

Liste d'attributs

UNL Specifications Version 3 Edition 1

Latest update 20 February 2003

ATTRIBUTE	DEFINITION	EXAMPLE
@ability	Ability, capability of doing something	The child can 't walk yet. He can speak English but he can't write it very well.
@admire	Admiring feeling of the speaker about something	
@affirmative	Sffirmation	
@although	Something follows against [contrary to] or beyond expectation	Although he didn't speak, I felt a certain warmth in his manner.
@abracket	< > is used, stand for "angle bracket"	

@begin	Beginning of an event or a state	It began to work again. work.@begin.@past
@blame	Blameful feeling of the speaker about something	A sailor, and afraid of the sea!
@brace	{ } is used	
@certain	Certainty that something is true or happens	If Peter had the money, he would have bought a car.
@complete	Finishing/completion of a (whole) event.	I've examined the script. examine.@complete
@conclusion	Logical conclusion due to a certain condition	He is her husband; she is his wife.
@confirmation	Confirmation	You won't say that, will you? It's red, isn't it? Then you won't come, right?
@consequence	Logical consequence	He was angry, therefore I left him alone.
@continue	Continuation of an event	He went on talking. talk.@continue.@past
@contrast	Contrasted UW	For instance, "but" in the examples below is used to introduce a word or phrase that contrasts with what was said before. It wasn't the red one but the blue one. He's poor but happy.
@custom	Customary or repetitious action	I used to visit there when I was a boy. visit.@custom.@past
@def	Already referred	The book you lost
@discontented	Discontented feeling of the speaker about something	(I'll tip you 10 pence.) But that's not enough!
@dissent	Dissenting feeling of the speaker about something	But that's not true.
@dparen	(( )) is used, stand for "double parenthesis"	
@dquote	" " is used, stand for "double quote"	
@emphasis	Emphasized UW	I do like it.
@end	Termination of an event or a state	I have done it. do.@end.@present
@entry	Entry or main UW of a sentence or a scope	He promised (entry of the sentence) that he would come(entry of the scope)
@exclamation	Feeling of exclamation	kirei na! ("How beautiful (it is)!" in Japanese) Oh, look out!
@expectation	Expectation of something	Children ought to be able to read by

		the age of 7. If you leave now, you should get there by five o'clock.
@experience	Experience	Have you ever visited Japan? visit.@experience.@interrogation I have been there. visit.@experience
@future	Will happen in future	He will arrive tomorrow
@generic	Generic concept	The dog is a faithful animal.
@grant	To give/get consent/permission to do something	Can I smoke in here? You may borrow my car if you like.
@grant-not	Not to give consent to do something	You {mustn't/are not allowed to/may not} borrow my car.
@imperative	Imperative	Get up! You will please leave the room.
@indef	Non-specific class	There is a book on the desk.
@inevitable	Logical inevitability that something is true or happens	There must be a mistake. They should be home by now.
@insistence	Strong will to do something	He will do it, whatever you say.
@intention	Intention about something or to do something	He shall get this money. (Speaker's intention) We shall let you know our decision.
@interrogative	Interrogation	Who is it?
@invitation	Inducement to do something	Will / Won't you have some tea? Let's go, shall we?
@just	Expresses an event or a state that has just begun or ended/been completed	He has just come. come.@complete.@just
@may	Practical possibility that something is true of happens	It may be true. It could be.
@need	Necessity of doing something	You need to finish thit work today.
@not	Complement set	Don't be late!
@obligation	Obligation to do something according to (quasi-) law, contract, or …	The vendor shall maintain the equipment in good repair.
@obligation-not	Obligation not to do something, forbid to do something according to (quasi-) law, contract or …	Cars must not park in front of the entrance. No smoking
@ordinal	Ordinal number	the 2nd door
@paren	( ) is used, stand for "parenthesis"	UNL (Universal Networking Language) cnt(UNL, Universal Networking Language.@parenthesis)

@past	Happened in the past	It was snowing yesterday
@pl	Plural	These (this.@pl) are the wrong size.
@present	Happening at present	It's raining hard.
@progress	An event is in progress	I am working now. work.@progress.@present
@polite	Polite feeling. Puts emphasis on a way of talking.	Could you (please)... If you could … I would …
@possible	Logical possibility that something is true or happens	Anybody can make mistakes. If Peter had the money, he would buy a car.
@probable	(Practical) probability that something is true or happens	That would be his mother. He must be lying.
@qfocus	Focused UW of a question	Are you painting the bathroom blue? To this question, the answer will be "No, I'm painting the LIVING-ROOM blue"
@rare	rare logical possibility that something is true or happens	If such a thing should happen, what shall we do? If I should fail, I will [would] try again.
@regret	regretful feeling of the speaker about something	It's a pity that he should miss such a golden opportunity.
@repeat	repetition of an event	It is so windy that the tree branches are knocking against the roof. knock.@entry.@present.@repeat
@request	request	Please don't forget…
@respect	respectful feeling. In many cases, some special words are used.	o taku ("(your) house" in Japanese) Good morning, sir.
@should	to do something as a matter of course	You should do as he says. You ought to start at once.
@squote	' ' is used, stand for "single quote"	
@sbracket	[ ] is used, stand for "square bracket"	
@state	Final state or the existence of the object on which an action has been taken	It is broken. break.@state
@surprised	Surprised feeling of the speaker about something	(He has succeeded!) But that's great!
@theme	Instantiates an object from a different class	
@title	Title	
@topic	Topic	He(@topic) was killed by her. The girl(@topic) was given a doll. This doll(@topic) was given to the girl.
@unreal	Unreality that something is true or happens	If we had enough money, we could buy a car.

		If Peter had the money, he could buy a car.
@vocative	Vocative	Boys, be ambitious!
@will	Will to do something	I'll write as soon as I can. We won't stay longer than two hours.
@wish	Wishful feeling, to wish something is true or has happened	If only I could remember his name! (~I `do wish I could remember his name!) You might have just let me know.
@yet	Expresses the feeling of something not yet begun, ended or completed, or expresses an event or a state that has not yet started or ended/been completed, together with @not.	I have not yet done it. do.@complete.@not.@yet

Annexe B : DTD et schéma d'UNL-xml

DTD d'UNL-xml

```
<!-- unl.dtd  this DTD conforms to UNL specifications V3.1 dated 20/05/2002. To be used in the UNL-xml file.
This schema is identified by the location:
 http://www-clips.imag.fr/geta/User/wang-ju.tsai/dataform/unl[1-1].dtd
$Author: tsai $ Wang-Ju TSAI Wang-Ju.Tsai@imag.fr
$Date: 2003/05/19 05:15:24 $
$Revision: 1.1 $
 -->
<!DOCTYPE D [
<!ELEMENT D (P+) >
<!ELEMENT P (S+)>
<!ELEMENT S (org,unl,GS+)>
<!ELEMENT org (arc+)>
<!ELEMENT unl (#PCDATA)>
<!ELEMENT GS (#PCDATA)>
<!ATTLIST D             dn      CDATA #REQUIRED
                        on      CDATA #REQUIRED
                        did     CDATA #IMPLIED
                        dt      CDATA #IMPLIED
                        mid     CDATA #IMPLIED>
<!ATTLIST P     number CDATA #REQUIRED>
<!ATTLIST S     number CDATA #REQUIRED>
<!ATTLIST org   lang    CDATA #REQUIRED
                code    CDATA #IMPLIED >
<!ATTLIST unl   sn      CDATA #IMPLIED
                pn      CDATA #IMPLIED
                rel     CDATA #IMPLIED
                dt      CDATA #IMPLIED
                mid     CDATA #IMPLIED>

<!ATTLIST GS    lang    CDATA #REQUIRED
                code    CDATA #IMPLIED
                sn      CDATA #IMPLIED
                pn      CDATA #IMPLIED
                rel     CDATA #IMPLIED
                dt      CDATA #IMPLIED
                mid     CDATA #IMPLIED>
]>

<!-- GS = generated sentence -->
<!-- dn = document name -->
<!-- on = owner name -->
<!-- did = document id -->
<!-- dt = date -->
<!-- mid = mail address -->
<!-- lang = lang tag -->
<!-- code = character code name -->
<!-- sn = system name -->
<!-- pn = post editor name -->
<!-- rel = reliability -->
```

schéma d'UNL-XML

Nous avons aussi défini UNL-xml au moyen d'un schéma XML. C'est plus verbeux, mais cela nous permet :

de mieux contrôler les attributs,

d'introduire l'espace de noms (UNL),

de définir ensuite des variantes pour différentes applications (comme UNL-xml-coéd déjà mentionné).

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
 XML Schema for a document UNL-XML.Comforms to UNL specifications V3.1 dated
20/05/2002.

 Namespace = http://www-clips.imag.fr/geta/User/wang-ju.tsai/dataform/
 Another mirror site is http://unlwjt.free.fr/dataform/

 This schema is identified by the location:
 http://www-clips.imag.fr/geta/User/wang-ju.tsai/dataform/unl[1-1].xsd

 $Author: tsai $ Wang-Ju TSAI Wang-Ju.Tsai@imag.fr
 $Date: 2003/05/19 05:15:24 $
 $Revision: 1.1 $

 Differences from previous version:
-name space added/
-change "reliability" range from [0-1] to [0-255]/
-comments

 -->
<xs:schema
 xmlns="http://www-clips.imag.fr/geta/User/wang-ju.tsai/dataform"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
 targetNamespace="http://www-clips.imag.fr/geta/User/wang-ju.tsai/dataform"
 elementFormDefault="qualified"
>
<!--  xmlns:xlink='http://www.w3.org/1999/xlink' -->


<xs:annotation>
  <xs:documentation xml:lang="en">
 UNL-XML schema for UNL specifications V3.1 Dated 20022002

 Namespace = http://www-clips.imag.fr/geta/User/wang-ju.tsai/dataform

 This schema is identified by the location:
 http://www-clips.imag.fr/geta/User/wang-ju.tsai/dataform/unl[1.0].xsd
  </xs:documentation>
</xs:annotation>



<!--   UNL-XML schema   -->


<xs:complexType name="D">
```

```
  <xs:sequence>
          <xs:element name="P" type="Ptype" minOccurs="1" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="number" type="numtype" use="required"/>
  <xs:attribute name="dn" type="xs:string" use="required"/>
  <xs:attribute name="on" type="xs:string" use="required"/>
  <xs:attribute name="did" type="xs:string" use="optional"/>
  <xs:attribute name="dt" type="xs:string" use="optional"/>
  <xs:attribute name="mid" type="xs:string" use="optional"/>
</xs:complexType>


<!--
dn= document name
on= owner name
did= document id
dt= date
mid= e-mail address
-->

<xs:complexType name="Ptype">
  <xs:sequence>
          <xs:element name="S" type="Stype" minOccurs="1" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="number" type="numtype" use="required"/>
</xs:complexType>

<xs:simpleType name="numtype" type="xs:positiveInteger"/>

<!--   sentence type, GSs=generated sentenses  -->

<xs:complexType name="Stype">
  <xs:sequence>
          <xs:element name="Org" type="orgtype" minOccurs="1" maxOccurs="1"/>
          <xs:element name="Unl" type="unltype" minOccurs="1" maxOccurs="1"/>
          <xs:element name="GSs" type="GStype" minOccurs="1" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="number" type="numtype" use="required"/>
</xs:complexType>

<!--   original text type   -->

<xs:complexType name="orgtype">
  <xs:sequence>
<xs:element name="org" type="xs:string" use="required" minOccurs="1" maxOccurs="1"/>
  </xs:sequence>
  <xs:attribute name="langcodedata" type="langcode" use="required"/>
</xs:complexType>

<xs:complexType name="langcode">
  <xs:sequence>
    <xs:element name="lang" use="required">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="ab"/>
          <xs:enumeration value="cn"/>
          <xs:enumeration value="de"/>
          <xs:enumeration value="el"/>
          <xs:enumeration value="es"/>
          <xs:enumeration value="fr"/>
```

```
            <xs:enumeration value="id"/>
            <xs:enumeration value="hd"/>
            <xs:enumeration value="it"/>
            <xs:enumeration value="jp"/>
            <xs:enumeration value="lv"/>
            <xs:enumeration value="mg"/>
            <xs:enumeration value="pg"/>
            <xs:enumeration value="ru"/>
            <xs:enumeration value="sh"/>
            <xs:enumeration value="th"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:element>
    <xs:element name="code" type="xs:string" use="optional"/>
  </xs:sequence>
</xs:complexType>



<!--   UNL code type   -->

<xs:complexType name="unltype">
  <xs:sequence>
          <xs:element name="arc" type="xs:string" minOccurs="1" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="uinf" type="inf" use="optional"/>
</xs:complexType>


<!--   information about the unl code   -->
<!--
sn= system name
pn= post editor name
rel= reliablity/degree of certainty (in the specs rel is in between 0 and 1, but in more recent
document, it seems to be between 0 and 255)
dt= date
mid= e-mail address
-->



<xs:complexType name="inf">
  <xs:sequence>
          <xs:element   name="sn"   type="xs:string"   use="required"   minOccurs="1"
          maxOccurs="1"/>
          <xs:element   name="pn"   type="xs:string"   use="required"   minOccurs="1"
          maxOccurs="1"/>
          <xs:element name="rel" type="relType" use="optional"/>
          <xs:element   name="dt"   type="xs:string"   use="optional"   minOccurs="0"
          maxOccurs="1"/>
          <xs:element   name="did"   type="xs:string"   use="optional"   minOccurs="0"
          maxOccurs="1"/>
  </xs:sequence>
</xs:complexType>


<xs:simpleType name="relType">
  <xs:restriction base="xs:integer">
    <xs:minInclusive value="0"/>
    <xs:maxInclusive value="255"/>
```

```
  </xs:restriction>
</xs:simpleType>

<!--   Generated sentence type   -->

<xs:complexType name="GStype">
  <xs:sequence>
          <xs:element name="GS" type="xs:string" minOccurs="1" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="Sinf" type="sinf" use="required"/>
</xs:complexType>

<!--   generated sentence information    -->

<xs:complexType name="sinf">
  <xs:sequence>
          <xs:element name="langcodegs" type="langcode" use="required"/>
          <xs:element name="infgs" type="inf" use="optional"/>
  </xs:sequence>
</xs:complexType>


</xs:schema>
```

Annexe C□ Corpus UNL

Exemple d'un document UNL-xml

```
<?xml version="1.0" ?>
- <!--
 <?xml-stylesheet type="text/xsl" href="newshow2.xsl"?>
  -->
- <!--
 XML pour UNL 20/02/2002
  -->
  <!DOCTYPE D (View Source for full doctype...)>
- <D dn="UNL News 2002021" on="RTM UNLCenter" dt="02/01/2002">
- <P number="1">
- <S number="3">
  <org lang="el">The First Conference on Building Global Knowledge (26-29 November 2001) concluded with the "Resolution in Suzhou".</org>
  <unl>obj(conclude(icl>end(obj>thing)).@entry.@past, :01)
 mod:01(conference(icl>meeting).@entry.@def, 1.@ordinal)
aoj:01(on(icl>about), conference(icl>meeting).@entry.@def)
obj:01(on(icl>about), build(agt>thing,obj>thing))
obj:01(build(agt>thing,obj>thing), knowledge(icl>information))
mod:01(knowledge(icl>information), global(icl>worldwide))
tim(:01, day(icl>date).@pl) tim(November, year(icl>date))
tim(day(icl>date).@pl, November) mod(year(icl>date), 2001)
mod(day(icl>date).@pl, :02) fmt:02(26.@entry,29)
man(conclude(icl>end(obj>thing)).@entry.@past, with(icl>how))
obj(with(icl>how(obj>thing)), :03.@double_quote)
plc:03(resolution(icl>decision).@entry.@def, Suzhou(icl>city))</unl>
  <GS lang="ab">_____ _____ __ _____ _____ _____ (26-29 _____ 2001) _____ _"_____ __ _____"</GS>
  <GS lang="el">The First Conference on Building Global Knowledge (26-29 November 2001) concluded with the "Resolution in Suzhou".</GS>
  <GS lang="cn">_____(26-29 2001 11_) _" __Suzhou " __</GS>
  <GS lang="es">la conferencia en saber global se construye de 1 en días en noviembre en año de 2001 de 26 a 29 se concluyó con la resolución en Suzhou.</GS>
  <GS lang="fr">La première conférence sur la construction de connaissance globale les jours 26-29 novembre 2001 s'est conclue avec la "résolution à Suzhou".</GS>
  <GS lang="hd">sArvaBOmika jFAna nirmANa para pahalA sammelana (26-29 navaMbara 2001) "sujZU prastAva" ke sAtha samApta huA.</GS>
  <GS lang="id">Konferensi pertama mengenai membangun pengetahuan global pada hari-hari dari 26 sampai 29 Nopember tahun 2001 telah menyimpulkan resolusi di Suzhou.</GS>
  <GS lang="it">La Prima Conferenza sulla Creazione della Conoscenza Globale nei giorni 26-29 Novembre 2001 si e' conclusa con la "Risoluzione di Suzhou".</GS>
```

< G S
lang="jp">_____
_“_____”_____</GS>
<GS lang="ru">_____ _____ «_____ _____
_____» (26 - 29 _____ 2001) _____ _____ «_____
_ _____».</GS>
</S>
- <S number="4">
<org lang="el">**The "Resolution in Suzhou" marks a turning point in the development of the UNL, both in terms of the strategic direction and in the management of its development and deployment.**</org>
<unl>**aoj(mark(icl>express).@entry, :01.@double_quote)
plc:01(resolution(icl>decision).@entry.@def, Suzhou(icl>city))
obj(mark(icl>express).@entry, turning point(icl>position))
plc(mark(icl>express).@entry, development(icl>activity):01.@def)
mod(development(icl>activity):01.@def, UNL(icl>Universal Networking Language).@def)
man(mark(icl>express).@entry, in terms of(icl>how(obj>thing)))
obj(in terms of(icl>how(obj>thing)), :02) mod(:02, both(mod<thing))
and:02(management(icl>activity).@entry.@def,
direction(icl>activity).@def)
mod:02(direction(icl>activity).@def, strategic(mod<thing))
mod:02(management(icl>activity).@entry.@def, :03)
and:03(deployment(icl>activity).@entry, development(icl>activity):02)
mod:02(:03, it)**</unl>
<GS lang="ab">_____ "_____ __ _____" _____ _____ __ _____ __**UNL**
_____ _____ _____ _____ _ _____ ___ _____ _
_____</GS>
<GS lang="el">**The "Resolution in Suzhou" marks a turning point in the development of the UNL, both in terms of the strategic direction and in the management of its development and deployment.**</GS>
< GS lang="cn" > " _ _ _ **Suzhou** " _____**UNL** __,
_____</GS>
<GS lang="es">**la resolución en Suzhou es marco en términos de amba la dirección estratégica y la gestión su de desarrollo y despliegue punto de inflexión en el desarrollo de U.N.L..**</GS>
<GS lang="fr">**La "résolution à Suzhou" marque un tournant dans le développement de l'UNL, en termes de direction stratégique et de gestion de son développement et de son déploiement.**</GS>
<GS lang="hd">**"sujZU prastAva" yU ena ela ke vikAsa meM, yuddhaanItika diSA Ora isake vikASa tathA pariniyojana ke prabaMdhana, donoM ke hI saMbaMdha meM eka saMkrAMti-kAla xarSAtA hE.**</GS>
<GS lang="id">**Resolusi di Suzhou menandai saat yang menentukan dalam pembangunan UNL, dalam hal strategis pengarahan dan manajemen dari pembangunan dan penyebarannya.**</GS>
< GS lang="it">**La "Risoluzione di Suzhou" segna una svolta nello sviluppo di UNL, nei termini della direzione strategica e della gestione del suo sviluppo e utilizzo.**</GS>
< G S
lang="jp">“_____”_____
_**UNL**_____</GS>
<GS lang="ru">«_____ _ _____» _____ _____ _____
_ _____ **UNL,** ___ _ _____ _____ _____ _____,
___ _ _ _____ _____ _____ _____ _ _____.</GS>
</S>

```
</P>
  </D>
```

Annexe D : Variables de PILAF et AUTOTAG

Table des catégories morphosyntaxiques de Pilaf

adv	Adverbe
subc	substantif commun
detp	Déterminant-pronom
det	Déterminant
subp	Substantif propre
adjq	Adjectif qualificatif
vide	
infi	Infinitif
ppt	Participe présent
ppas	Participe passé
verb	Verbe
xet	Auxiliaire être
xav	Auxiliaire avoir
pnt	Point
dpnt	Deux points
virg	Virgule
coco	Conjonction de coordination
prep	Préposition
locp	Locution préposition
cocs	Conjonction subordination
locs	Locution conjonctive
padv	Pronom adverbial y en
prl	Pronom relatif sauf
prlc	Pronom relatif conjonctif
ne	Négation ne
pas	2ème négation pas
ce	Pronom démonstratif
pper	Pronom personnel
phra	Phrase
vet	Verbe être
pnp	Pronom non personnel
loca	Locution adverbiale
ppf	Pronom personnel fort
adji	Adjectif indéfini
chf	Chiffre
date	Date
intj	Interjection
quan	Quantification
vav	Verbe avoir
de	Préposition de
à	prep et à la (au)
cls	SuperCl

Table des variables morphologiques de Pilaf

fem	Féminin
mas	Masculin
ind	Indicatif
cdl	Conditionnel
sub	Subjonctif
imp	Impératif
pre	Présent
fut	Futur
imi	Imparfait
pas	Passé simple
sin	Singulier
plu	Pluriel
uno	Première personne
dos	Deuxième personne
tre	Troisième personne
ref	Réfléchi
ide	Indéfini
pat	Partitif

Variables syntaxiques

Bien que PILAF soit un analyseur morphologique, il est capable de donner quelques valeurs des variables syntaxiques. Pourtant le résultat n'est pas sans ambiguïté.

suj	Sujet (comme *je, tu*)
cod	Complément d'objet direct (*le, la*)
dat	Complément d'objet indirect (*lui, leur*)
cpr	Complément prépositionnel (à *quoi*)
cdn	Complément de nom (livre *de Marie*)

Exemple de sortie de PILAF

Nous avons déjà montré la sortie de PILAF sur le site web. Ici nous montrons la sortie de PILAF de la version texte. Nous avons téléchargé PILAF et l'avons intégré dans notre maquette de coédition. Sinon, PILAF peut être appelé par une requête CGI à travers Internet.

On envoie à Pilaf : « L'agence promeut la participation économique et politique d'une femme »

Le résultat est une chaîne. Le symbole « [* » signifie le début d'un mot. Le symbole « *!* » signifie la fin d'un mot. Chaque lemme candidat et ses informations grammaticales est séparé par un « * ».

Retour de Pilaf élagué :

[*l' l' detp sin fem mas tre cod*!*[*agencer agence verb sin dos imp*agencer agence verb sin tre uno pre sub*agencer agence verb sin tre uno pre ind*agence agence subc sin fem*!*[*promouvoir promeut verb sin tre pre ind*!*[*la la detp sin fem tre cod*!*[*participation participation subc sin fem*!*[*économique économique adjq sin fem mas*!*[*et et coco*!*[*politique politique subc sin fem mas*politique

politique adjq sin fem mas*!*[*d' d' prep*d' d' det plu fem mas ide pat*!*[*une une det sin fem ide*une une subc sin fem*!*[*femme femme subc sin fem*!*

Table de catégories du chinois moderne (utilisé par «AUTOTAG»)

Catégories Standard	Catégories du chinois moderne	Explication
A	A	épithète
ADV	D ,Da ,Dfa ,Dfb ,Dk	adverbe
ASP	Di	particule d'aspect
C	Caa ,Cbb	conjonction
DET	Nep ,Neqa ,Nes ,Neu	déterminant
FW	FW	mot étranger
M	Nf	spécificatif
N	Na ,Nb ,Nc ,Ncd ,Nd ,Nh	nom
P	P	préposition
POST	Cab ,Cba ,Neqb ,Ng	postposition
T	DE ,I ,T	particule
Vi	VA ,VB ,VH ,VI	verbe intransitif
Vt	SHI ,VAC ,VC ,VCL ,VD ,VE ,VF , VG ,VHC ,VJ ,VK ,VL ,V_2	verbe transitif
NAV		prédicat nominal

Table de catégories du segmenteur AUTOTAG

Catégorie du chinois moderne	Explication	
A		épithète
D		adverbe
Da		adverbe de quantité
Dfa		adverbe de degré devant verbe
Dfb		adverbe de degré après verbe
Dk		adverbe phrastique
Di		particule d'aspect
Caa		conjonctif « et » »avec »
Cbb		corrélatif
Nep		déterminant relatif
Neqa		déterminant de quantité
Nes		déterminant spécial
Neu		déterminant cardinal
FW		mot étranger
Nf		spécificatif
Na		nom commun
Nb		nom propre
Nc		nom géographique
Ncd		nom de localisation
Nd		nom de temps
Nh		pronom
P		préposition
Cab		conjonctif comme « etc. »
Cba		conjonctif comme « si »
Neqb		numéral postpositionnel
Ng		postposition
DE	_, _, _, _	particule « de »
I		interjection
T		auxiliaire
VA		verbe intransitif d'action
VB		verbe transitif d'action (I)
VH		verbe intransitif d'état
VI		verbe transitif d'état (I)
SHI	_	verbe « shi (être) »
VAC		verbe impératif d'action
VC		verbe transitif d'action (II)
VCL		verbe à l'objet de lieu

VD	____	verbe à deux objets
VE	_____	verbe d’action à l’objet phrasique
VF	_____	verbe d’action à l’objet prédicatif
VG	____	verbe de catégorisation
VHC	_____	verbe impératif d’état
VJ	_____	verbe transitif d’état (II)
VK	_____	verbe d’état à l’objet phrasique
VL	_____	verbe d’état à l’objet prédicatif
V_2	_	verbe « you (avoir) »

Annexe E : Page extraite du dictionnaire unl-geta_fr_unl.unl

[Aaron]{CAT(CATN),GNR(MAS),N(NP)}"Aaron(icl>human,fld>religion)";
[abaissant]{CAT(CATADJ)}"degrading(icl>state)";
[abaisse]{CAT(CATN),GNR(FEM),N(NC)}"rolled-out_pastry(fld>food)";
[abaisse-langue]{CAT(CATN),GNR(MAS),N(NC)}"spatula(fld>medicine)";
[abaisseur]{CAT(CATN),GNR(MAS),N(NC)}"depressor(icl>muscle)";
[abaque]{CAT(CATN),GNR(MAS),N(NC)}"abacus(icl>tool)";
[abaque]{CAT(CATN),GNR(MAS),N(NC)}"graph";
[abasourdissement]{CAT(CATN),GNR(MAS),N(NC)}"bewilderment";
[abâtardir]{AUX(AVOIR),CAT(CATV),VAL1(GN)}"make_degenerate(icl>state)";
[abâtardissement]{CAT(CATN),GNR(MAS),N(NC)}"debasement";
[abats]{CAT(CATN),GNR(MAS),N(NC),NUM(PLU)}"offal";
[abattage]{CAT(CATN),GNR(MAS),N(NC)}"felling(mod>tree)";
[abattage]{CAT(CATN),GNR(MAS),N(NC)}"slaughter(mod>animal)";
[abattant]{CAT(CATN),GNR(MAS),N(NC)}"leaf(mod>desk)";
[abattant]{CAT(CATN),GNR(MAS),N(NC)}"lid(mod>WC)";
[abattis]{CAT(CATN),GNR(MAS),N(NC),NUM(PLU)}"giblets(fld>culinary)";
[abattis]{CAT(CATN),GNR(MAS),N(NC),NUM(PLU)}"limbs";
[abat-vent]{CAT(CATN),GNR(MAS),N(NC)}"cowl(mod>chimney)";
[abat-vent]{CAT(CATN),GNR(MAS),N(NC)}"wind_break";
[abbatial]{CAT(CATADJ)}"abbey(fld>religion)";
[abbatiale]{CAT(CATN),GNR(FEM),N(NC)}"abbey_church(fld>religion)";
[abbesse]{CAT(CATN),GNR(FEM),N(NC)}"abbess(fld>religion)";
[Abdias]{CAT(CATN),GNR(MAS),N(NP)},"obadiah";
[abdominal]{CAT(CATADJ)}"abdominal";
[abdominaux]{CAT(CATN),GNR(MAS),N(NC),NUM(PLU)}"abdominal_muscles(icl>muscles)";
[abducteur]{CAT(CATN),GNR(MAS),N(NC)}"abductor";
[abécédaire]{CAT(CATN),GNR(MAS),N(NC)}"spelling_book";
[Abel]{CAT(CATN),GNR(MAS),N(NP)}"abel(fld>religion,icl>human)";
[abêtissant]{CAT(CATADJ)}"mindless(icl>state)";
[abêtissement]{CAT(CATN),GNR(MAS),N(NC)}"stupefying_effect(icl>event)";
[abêtissement]{CAT(CATN),GNR(MAS),N(NC)}"mindlessness(icl>state)";
[Abidjan]{CAT(CATN),GNR(MAS),N(NP)}"Abidjan(icl>town)";
[abiotique]{CAT(CATADJ)}"abiotic";
[abjectement]{CAT(CATADV)}"despicably(icl>manner)";
[abjection]{CAT(CATN),GNR(FEM),N(NC)}"abjectness";
[abjuration]{CAT(CATN),GNR(FEM),N(NC)}"abjuration";
[ablatif]{CAT(CATN),GNR(MAS),N(NC)}"ablative(icl>linguistics)";
[ablette]{CAT(CATN),GNR(FEM),N(NC)}"bleak(fld>zoology)";
[ablution]{CAT(CATN),GNR(FEM),N(NC)}"ablution(icl>religion)";
[ablutions]{CAT(CATN),GNR(FEM),N(NC),NUM(PLU)}"ablutions";
[abolitionnisme]{CAT(CATN),GNR(MAS),N(NC)}"abolitionism(icl>doctrine)";
[abolitionniste]{CAT(CATN),GNR(MAS,FEM),N(NC)}"abolitionist(icl>human)";
[abominablement]{CAT(CATADV)}"abominably(icl>manner)";
[abomination]{CAT(CATN),GNR(FEM),N(NC)}"abomination";
[abominer]{AUX(AVOIR),CAT(CATV),VAL1(GN)}"abominate";
[abortif]{CAT(CATADJ)}"abortive(icl>state)";
[abortif]{CAT(CATN),GNR(MAS),N(NC)}"abortifacient";
[aboucher]{AUX(AVOIR),CAT(CATV),VAL1(GN)}"butt(fld>technology,icl>event)";
[Abou_Dhabi]{CAT(CATN),GNR(MAS),N(NP)}"abu_dhabi";
[abouler]{AUX(AVOIR),CAT(CATV),VAL1(GN)}"hand_over(agt>human,icl>event)";
[aboulie]{CAT(CATN),GNR(FEM),N(NC)}"abulia";
[aboulique]{CAT(CATADJ)}"abulic";
[aboulique]{CAT(CATN),GNR(MAS,FEM),N(NC)}"person_suffering_from_abulia";

[Abou_Simbel]{CAT(CATN),GNR(MAS),N(NP)}"abu_simbel";
[about]{CAT(CATN),GNR(MAS),N(NC)}"end(icl>building)";
[aboutement]{CAT(CATN),GNR(MAS),N(NC)}"butt_jointing(icl>process)";
[abouter]{AUX(AVOIR),CAT(CATV),VAL1(GN)}"butt(icl>event,obj>thing)";
[abouti]{CAT(CATADJ)}"accomplished(icl>state)";
[aboutissants]{CAT(CATN),GNR(MAS),N(NC),NUM(PLU)}"outs";
[aboyeur]{CAT(CATN),GNR(MAS),N(NC)}"usher(icl>human)";
[abracadabra]{CAT(CATN),GNR(MAS),N(NC)}"abracadabra";
[Abraham]{CAT(CATN),GNR(MAS),N(NP)}"Abraham(fld>religion,icl>human)";
[abraser]{AUX(AVOIR),CAT(CATV),VAL1(GN)}"abrade(icl>event)";
[abrasion]{CAT(CATN),GNR(FEM),N(NC)}"abrasion";
[abrègement]{CAT(CATN),GNR(MAS),N(NC)}"shortening";
[abricoté]{CAT(CATADJ)}"apricot-flavoured(icl>state)";
[abricotier]{CAT(CATN),GNR(MAS),N(NC)}"apricot_tree(icl>plant)";
[abrité]{CAT(CATADJ)}"sheltered(icl>state)";
[abruptement]{CAT(CATADV)}"steeply(icl>manner)";
[abruptement]{CAT(CATADV)}"suddenly";
[abrutissant]{CAT(CATADJ)}"deafening(icl>state)";
[abrutissement]{CAT(CATN),GNR(MAS),N(NC)}"mindless_state";
[ABS]{CAT(CATN),GNR(MAS),N(NC)}"ABS(fld>technology,icl>vehicle)";
[abscisse]{CAT(CATN),GNR(FEM),N(NC)}"abscissa(fld>mathematics)";
[abscons]{CAT(CATADJ)}"abstruse(icl>state)";
[absenter_(s')]{AUX(AVOIR),CAT(CATV),REFLEX(1)}"go_away(agt>human,icl>event)";
[abside]{CAT(CATN),GNR(FEM),N(NC)}"apse(fld>architecture)";
[absidial]{CAT(CATADJ)}"apsidal(icl>state)";
[absidiole]{CAT(CATN),GNR(FEM),N(NC)}"apsidiole(fld>architecture)";
[absolutisme]{CAT(CATN),GNR(MAS),N(NC)}"absolutism(icl>doctrine)";
[absolutiste]{CAT(CATN),GNR(MAS,FEM),N(NC)}"absolutist(icl>human)";
[absoute]{CAT(CATN),GNR(FEM),N(NC)}"absolutions";
[abstentionnisme]{CAT(CATN),GNR(MAS),N(NC)}"abstentionism(icl>doctrine)";

Annexe F□ Exemple complet de planche de grammaire statique

Langue décrite: français

Grammaire statique: Fexemple

Planche n°: 17

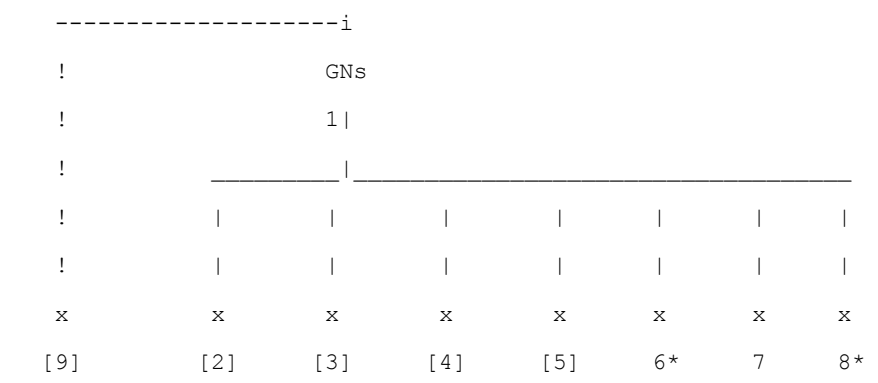
Type: GROUPE NOMINAL (SIMPLE)

Cas traité: Groupe nominal gouverné par un nom commun

Planches référées: 8,9,10 (GAs)

15 (coordination de GAs)

ZONE 1



K.		ADVN		GCARD	GA		GA
cat/scat.	s/prep		d/-			n/nc	
UL…							
SF	REG	ATG	DES	QTF	ATG	GOV	ATG

ZONE 2

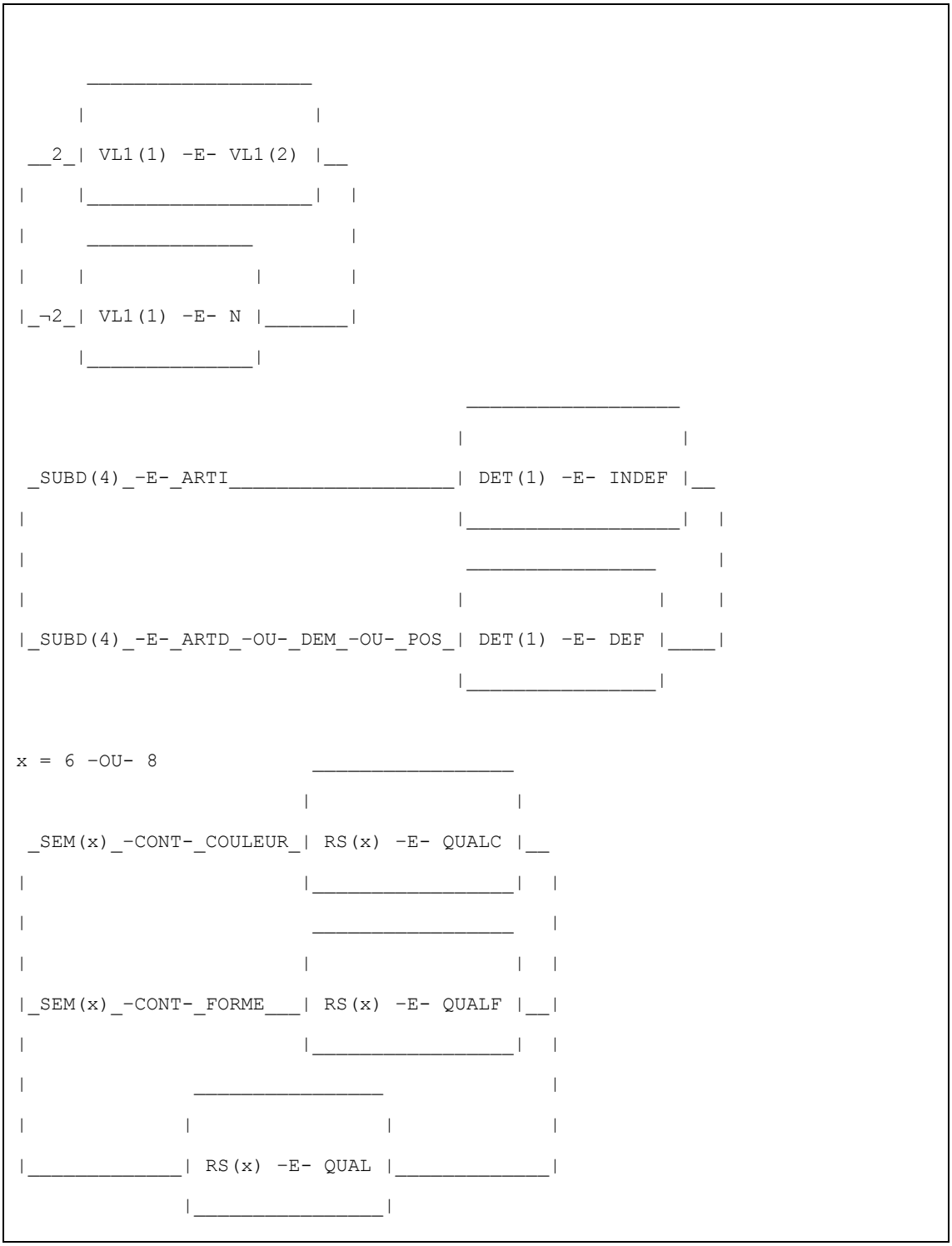
3 –IMP– 4 –OU– 5

(9 –ET– ¬2) –IMP– (4 –OU– 5)

$AGR.GNR.NBR(4,5,6,7,8)

ZONE 3

GNR.NBR(1) –E– GNR.NBR(4) –E– GNR.NBR(5) –E– GNR.NBR(6) –E– GNR.NBR(7) –E– GNR.NBR(8)


VL1(1) -E- VL1(2)
VL1(1) -E- N
SUBD(4)-E-_ARTI
DET(1) -E- INDEF
SUBD(4)-E-_ARTD_-OU-_DEM_-OU-_POS_
DET(1) -E- DEF
x = 6 -OU- 8
SEM(x)-CONT-_COULEUR_
RS(x) -E- QUALC
SEM(x)-CONT-_FORME___
RS(x) -E- QUALF
RS(x) -E- QUAL

Annexe H□ Exemple complet de l'ILT de KBMT-89

Voici un exemple d'expression de l'ILT de KBMT-89. La phrase japonaise d'entrée est la suivante :

_____
　　“　”
___ __ _____

(kaku souti no setuzoku ga syuuryou si ta ra sisutemu yunitto to purin-taa no den-gen-suitti ga “kiru” gawa ni natte iru koto okakunin-si te kudasai.)

La traduction anglaise de cette phrase est :

Confirm that the power unit switches of the system unit and the printer are in the “off” position when the connection of each device is complete.

L'étoile « * » marque une référence à un concept dans le modèle de domaine, et l'esperluette « & » marque une référence à un ensemble de valeurs dans le modèle de domaine.

```
(make-frame-old clause1
        (ilt-type (value clause))
        (clauseid (value clause1))
        (propositioned (value proposition1))
        (discourse-cohesion-marker (value (conditional clause2)))
        (speechactid (value speech-act1))
)

(make-frame-old proposition1
      (ilt-type (value proposition))
      (propositionid (value proposition1))
      (clauseid (value clause1))
      (aspectid (value apsect1))
      (complete (value yes))
      (is-token-of (value *connect))
      (agent (value unknown))
      (theme (value role2))
      (time (value time1))
      )

(make-frame-old role2
      (ilt-type (value role))
```

```
      (clauseid (value clause1))
      (is-token-of (value *device))
      (r-quantifier (value universal))
      (reference (value definite))
      )

(make-frame-old aspect1
      (ilt-type (value aspect))
      (clauseid (value clause1))
      (phase (value end))
      )

(make-frame-old speech-act1
      (ilt-type (value speech-act))
      (speech-act (value statement))
      (direct? (value yes))
      (speaker (value author))
      (hearer (value reader))
      (time (value (before time1)))
      )

(make-frame-old clause2
      (ilt-type (value clause))
      (clauseid (value clause2))
      (propositioned (value proposition2))
      (speechactid (value speech-act2))
      )

(make-frame-old proposition2
      (ilt-type (value clause))
      (propositioned (value proposition2))
      (clauseid (value clause2))
      (aspect (value aspect2))
      (is-token-of (value *confirm))
      (agent (value role3))
      (theme (value clause3))
      (time (value (after time1)))
      )

(make-frame-old role3
      (ilt-type (value role))
      (clauseid (value clause2))
      (is-token-of (value *reader))
      (reference (value definite))
      )

(make-frame-old aspect2
      (ilt-type (value aspect))
      (clauseid (value clause2))
      (phase (value end))
      )

(make-frame-old speech-act2
      (ilt-type (value speech-act))
      (speech-act (value command))
      (direct? (value yes))
```

```
      (speaker (value author))
      (hearer (value reader))
      (time (value (before time1)))
      )

(make-frame-old clause3
      (ilt-type (value clause))
      (clauseid (value clause3))
      (propositioneid (value proposition3))
      (speechactid (value speech-act3))
      )

(make-frame-old proposition3
      (ilt-type (value proposition))
      (propositioned (value proposition3))
      (clauseid (value clause3))
      (aspect (value aspect3))
      (is-token-of (value *discrete-position))
      (range (value off-position))
      (domain (value role4))
      (time (value (after time1)))
      )

(make-frame-old role4
      (ilt-type (value role))
      (clauseid (value clause3))
      (is-token-of (value *set))
      (member (value *power-switch))
      (belongs-to (value roles5))
      )

(make-frame-old role5
      (ilt-type (value role))
      (clauseid (value clause3))
      (is-token-of (value *set))
      (member (value *system-unit *printer))
      (type (value conjunction))
      )

(make-frame-old aspect3
      (ilt-type (value aspect))
      (clauseid (value clause3))
      (phase (value end))
      )

(make-frame-old speech-act3
      (ilt-type (value speech-act))
      (speech-act (value command))
      (direct? (value yes))
      (speaker (value author))
      (hearer (value reader))
      (time (value(before time1)))
      )
```