

Thèse Professionnelle

Master Bases de Données et Progiciel

L5G et Web Sémantique

Auteur :

Julien Dallier

Etudiant au Master BDP - Ceram Sophia Antipolis

Tuteur technique :

M. Buffa

**Maître de conférence en informatique - Université de Nice Sophia Antipolis, ESSI,
INRIA**

Tuteur méthodologique :

M. Augier

Professeur de Technologies de l'Information - CERAM Sophia Antipolis

CERAM Sophia Antipolis
European School of Business



Avant-propos

Ce travail s'inscrit dans le programme du Master **Ms-BDP** que j'ai préparé sous forme de Badges au **CERAM de Sophia-Antipolis**.

Il s'agit d'une thèse professionnelle réalisée sous la double tutelle du **CERAM** et de la **Conférence des Grandes Ecoles**.

La thèse professionnelle du Master Ms-BDP doit répondre à quatre critères :

- x Un travail de réflexion personnelle et non un simple rassemblement d'informations déjà disponibles ;
- x Une thèse professionnelle, c'est à dire un travail sur un aspect concret de la vie des entreprises et non une recherche fondamentale de type universitaire ;
- x Une thèse susceptible d'intéresser un large public de dirigeants et non la monographie d'une seule entreprise ;
- x Un travail de haut niveau tant sur le fond que sur la forme, susceptible de contribuer au renom du Master Ms-BDP.

La thèse représente donc un véritable travail de recherche reposant sur le développement d'idées argumentées avec pertinence.

J'ai pour ma part choisi de m'interroger sur le concept de génération des langages de programmation avec une étude plus particulière de la dernière de ces générations : la cinquième.

Ce qui m'amènera, pour rester concret, à étudier l'ouverture qu'apporte cette génération de langage principalement dans le monde de l'Internet : la sémantique par l'utilisation de règles.

Ma thèse s'adresse donc à toute personne, dirigeant ou société, qui dans un souhait de perpétuelle veille technologique s'interroge sur l'état de l'art du monde de la programmation et plus précisément en ce qui concerne les grandes directions que prennent les développements pour l'Internet.

Il s'agit d'un travail de réflexion personnelle qui concerne un aspect concret et actuel de l'informatique et des traitements de l'information plus précisément, et qui est susceptible d'intéresser le plus large public.

Ce document à été rédigé sur un PC sous ® MS.Windows XP Pro en utilisant l'éditeur Texte de ® OpenOffice.org.

Copyright (c) 2005 Julien Dallier.

Permission est accordée de copier, distribuer et/ou modifier ce document selon les termes de la Licence de Documentation Libre GNU (GNU Free Documentation licence).

Une copie de la présente Licence est disponible à l'adresse :

<http://www.gnu.org/copyleft/fdl.html>.

Remerciements

Je tiens à remercier les personnes suivantes pour leur implication dans la bonne réussite personnelle et professionnelle que représente à mes yeux les formation MS BDP du Ceram, ainsi que pour leur contribution à la réalisation de ce document.

- x **Gabriel Mopolo Moke**, Ancien responsable des enseignements du Master BDP au Ceram Sophia Antipolis, que je remercie pour son implication, ses enseignements et ses conseils.
- x **Florence Tressols**, Gérante de la SARL **Hypsenoïa Interactive** (mon ancien employeur, société spécialisée dans la réalisation de solutions de communications innovant es), au sein de laquelle j'ai entamé cette formation, que je remercie pour tout, vraiment tout
- x **L'ensemble du corps enseignant** qui nous a suivi durant toute la période de formation, pour son efficacité et son accueil.(Merci Gabriel, Anthony, Dominique et puis tout le monde...)
- x **L'ensemble des responsables administratifs** qui nous ont suivi durant toute la période de formation et particulièrement Pascale Louis Girard, Alexandra Malialin, Evelyne Farrara et Michel Bernasconi.
- x Mon tuteur technique **Michel Buffa** et mon tuteur méthodologique, **Marc Augier** , pour nos conversations téléphoniques, nos échanges de courriers électroniques, ainsi que pour l'efficacité et la pertinence de leurs conseils
- x **Les filles d' Internénettes** (<http://www.internenettes.fr>) Elles m'ont tout simplement inspiré la rédaction de ce document... Donc c'est la moindre des choses (cf.[Conclusion](#) et [Annexe 5](#))
- x et ma chérie Anne Laure bien évidemment, pour son soutien.

Table des matières

Avant-propos.....	2
Remerciements.....	4
Introduction.....	6
1. Les différentes générations des langages de programmation.....	8
1.1. Petit historique.....	8
1.2. Leurs évolutions syntaxiques et sémantiques.....	9
1.3. Un exemple de langage de 5ème génération : Prolog.....	11
2. La 5ème génération de langage.....	14
2.1. Langage naturel et recherche d'informations.....	14
2.2. Les perspectives en recherche d'informations	15
2.3. Les technologies de règles.....	15
3. Lien entre la 5ème génération des langages et Web Sémantique.....	16
3.1. La sémantique, le message	16
3.2. De la mémoire collective vers le Web Sémantique.....	20
3.3. De la sémantique des textes au Web Sémantique.....	23
4. Le Web Sémantique.....	28
4.1. La spécification RDF.....	28
4.2. Les actions du W3C autour d'un langage.....	31
4.3. Des exemples et des applications.....	34
4.3.1 Le laboratoire de Web Sémantique (LabWebSem)	34
4.3.2 Triple.....	35
4.3.3 Dans l'entreprise.....	35
Conclusion.....	37
Références.....	39
Bibliographie chronologique.....	40
Annexes.....	43

Introduction

« En un mot, l'esprit, en plus de la syntaxe, a une sémantique. La raison pour laquelle un programme d'ordinateur ne sera jamais pareil à un esprit est tout simplement que le programme est purement syntaxique, tandis que l'esprit a quelque chose en plus. L'esprit est sémantique, au sens où, en plus de sa structure formelle, il a un contenu. »

John Rogers Searle Professeur de Philosophie (Université de Californie)

Que ce soit dans le monde de l'entreprise, dans le domaine politique, dans le cadre des loisirs ou encore de l'éducation, la communication est omniprésente.

L'essor des NTIC (Nouvelles Technologies de l'Information et de la Communication) ainsi que les innovations continues dans les langages de programmation sont aujourd'hui indéniables.

L'obligation actuelle de communiquer, de publier, d'informer se traduit notamment par le développement exponentiel d'Internet et du nombre de ses usagers (1).

De plus, le profil même des utilisateurs a évolué : les « experts » (ingénieurs, chercheurs, étudiants) ne sont plus les seuls à « surfer ».

Face à ce flot d'informations, des outils de recherche (moteurs, annuaires...) permettent aux utilisateurs de chercher et, idéalement, de trouver l'information souhaitée. Comme le soulignent Jean-Paul et Marie-Christine Haton (Haton 1993), une langue naturelle est un moyen privilégié d'interaction entre l'homme et la machine. Pouvoir poser une question en langage naturel à une base de données constitue un progrès notable pour l'utilisateur.

En ce sens, ce document est destiné à vous informer sur la tendance actuelle des nouveaux langages prévus pour produire des codes informatiques compréhensibles, voir naturels... La 5ème génération de langage de programmation dans le monde de l'informatique et les concepts qu'elle apporte plus particulièrement dans le sous-domaine de l'informatique que représente Internet.

Pour cela, je vais m'appliquer à définir les différentes générations de langages de programmation, les principales différences entre elles ainsi que quelques exemples, pour tenter de comprendre les enjeux apportés en cela dans le monde de l'Internet. Une étude de la notion de Web Sémantique, sera présentée en faisant le lien entre la 5ème génération des langages et le concept de l'analyse sémantique. Enfin je vous présenterai par le biais de l'étude d'exemples, d'articles et de normes les grandes nouveautés et perspectives engendrées par cette nécessité absolue de « donner du sens » à des contenus...

(1). Si en 1993, par exemple, la toile comportait quelques milliers de pages accessibles, on en compte aujourd'hui plus de 350 millions avec un taux de croissance estimé à 20 millions de pages par mois (Bourdoncle 1999). Par ailleurs, en France, l'usage d'Internet a progressé de 45% durant l'année 1999. Dans le même temps, le nombre d'Allemands utilisant Internet chez eux a doublé.

1. Les différentes générations des langages de programmation

1.1. Petit historique

Les langages de programmation sont à l'origine d'un besoin qui consiste à fournir des informations à une machine.

Le langage de programmation est un formalisme généralement textuel qui est compréhensible par l'homme (celui qui écrit ou gère le code source) et traduisible par un logiciel (le compilateur ou l'interpréteur) en une suite d'instructions compréhensibles par la machine (le langage machine).

On peut distinguer les langages par différents aspects :

Leur mode d'exécution :

- x **interprétés** : les instructions du code sources sont examinées à l'exécution puis traduites en langage machine
- x **compilés** : la traduction de l'interprétation est effectuée à l'avance, de sorte que le programme exécuté devient un ensemble d'instructions pour la machine
- x **virtuels** : à la fois compilés pour et interprétés par une machine virtuelle (Virtual Machine ou VM), c'est-à-dire une machine simulée par un logiciel ré écrit pour différentes machines. L'objectif de cette virtualité est la portabilité et ce qu'elle implique (indépendance des contraintes d'un OS, d'une machine, optimisations dynamiques).

Leur paradigme :

- x **procédural** : qui se focalise sur le verbe : *ouvre (porte)*.
On se préoccupe alors plus des événements qui peuvent survenir dans une application, et de comment ils doivent survenir.
- x **orienté objet (OO)** : qui se focalise sur l'objet : *porte.ouvre()*.
On se préoccupe alors plus des entités existantes et de leurs relations et interactions.
- x **hiérarchique**

Leur domaine d'application :

Langage de programmation, de requêtes ou de représentation.

Leur niveau d'abstraction :

Qui peut indifféremment être élevé (langages fonctionnels), moyen (langages OO), faible ou nul.

Leur génération :

1ère : langage machine (suite de bits)

2ème : assembleur (mnémoniques, manipulation de registres)

3ème : procédural

4ème : automatisation

5ème : interprétation du sens (langue naturelle, intelligence artificielle)

Note : Une représentation plus précise de ces générations sera faite par la suite.

Indépendamment de ces aspects, la conception d'un langage peut se définir par des **mots-clés** ou **mots réservés**, utiliser pour représenter des déclarations, des affectations, des structures de contrôles, des tests (if), des boucles (for, while) ou des branchements (goto, appel de méthode)

1.2. Leurs évolutions syntaxiques et sémantiques

Les langages de programmation généraux ont connu une évolution laborieuse depuis 1946. Pour montrer l'évolution de la syntaxe, des fonctionnalités, les voici classés par date d'implémentation en [Annexe 3](#)

Quand Pascal, C++, Python, sont apparus, qui les a imaginés, pourquoi les langages ont-ils ces différences ?

Les principales catégories de langages à ce jour, sont les langages fonctionnels et procéduraux (dits aussi impératifs), et les langages logiques.

Un langage est dit fonctionnel, au sens mathématique du mot fonction, si chaque opération est indépendante du contexte, et si le résultat d'une fonction dépend exclusivement de ses arguments.

Haskell, Lisp, sont des langages fonctionnels de genres différents. Ils sont plutôt destinés à l'intelligence artificielle. Lisp utilise le principe de réduction de problème, Prolog le principe de résolution par des prédicats de premier ordre, c'est un langage logique (c.f. [1.3. Un exemple de langage de 5ème génération : Prolog](#)).

On oppose aussi le style impératif au style déclaratif. En fait on considère déclaratif un système qui énonce des connaissances, l'énoncé d'un problème, et fournit un

mécanisme de résolution. Prolog se dit déclaratif et c'est en quoi il nous intéresse.

Pratiquement tous les langages, fonctionnels ou impératifs sont maintenant orientés objets, et utilisent des classes décrivant des objets réels ou purement informatiques.

Ces évolutions ainsi que certaines sources m'amènent finalement à distinguer 4 générations de langages de programmation et deux nouvelles tendances.

1ère : Langage machine (suite de bits)

2ème : Langages symboliques et auto codes assembleur (mnémoniques, manipulation de registres)

3ème : Langages procéduraux indépendants du matériel

4ème : Langages conçus pour décrire le problème, comme Simula et autres langages à objets, automatisation

Indépendamment de ces générations théoriques, quelques grandes dates permettent d'y voir un peu plus clair :

- x Années 50: Création des langages de haut niveau (plus proches de l'homme).
- x Années 60: Foisonnement de langages spécialisés. Forth. Simula I. Lisp, Cobol. On essaie sans succès d'imposer des langages généraux: Algol, PL/1.
- x Années 70: Duel entre la programmation structurée avec Pascal et l'efficacité du langage C (cela dure encore en 2000). Généralisation du Basic interprété sur les micro-ordinateurs apparus en 1977, jusqu'à la fin des années 80.
- x Années 80: Expérimentation d'autres voies et notamment des objets. ML. Smalltalk. Sur les micro-ordinateurs, on utilise maintenant C, Pascal, Basic compilé.
- x Années 90: Généralisation de la programmation objet grâce aux performances des micro-ordinateurs. Java, Perl, Python s'ajoutent aux langages micros.
- x Années 2000: Programmation Internet (et les innovations à venir).

Les nouvelles tendances :

Les langages à programmation logique qui prétendent représenter la cinquième génération via l'interprétation du sens (langue naturelle, intelligence artificielle)

La cinquième génération pourrait être celle des langages Internet, c'est à dire fonctionnant sur toute machine et compilés en code intermédiaire (dit virtuel).

Les langages « Markup » inspirés de xml sont la dernière tendance, ils intègrent le code et les données sous une forme extensible, et qui fonctionnent sur le Web..

Une analyse plus poussée des différents langages dans l'[annexe 2.1](#) démontre qu'après la pléthore de dialectes des années 70, l'invention de langages a stagné quant à la syntaxe. Les langages courants, même récents comme Java, C#, PHP, n'apportent aucun changement aux instructions du traitement.

Les capacités actuelles des ordinateurs ne seraient-elles pas entièrement exploitées ?

La plateforme .Net pas exemple permet de faciliter l'intégration de code à l'intérieur des données, mais le Xml peut aussi être une alternative. Même si le C# tend à se populariser, ce sera grâce à la force de l'habitude, et comme successeur du C++ et de Java.

.NET, en permettant d'utiliser des langages différents avec les bibliothèques existantes, devrait favoriser l'apparition de nouveaux langages, plus intéressants parce que plus proches de la pensée, beaucoup plus que ne le sont les langages actuels.

La plateforme .Net utilise Xml en le convertissant en code orienté-objet. L'avenir est plutôt à l'utilisation de Xml directement comme structure de données.

D'autres tendances apparaissent dans les langages avec la programmation par aspects, ou par schémas tel UML...

1.3. Un exemple de langage de 5ème génération : Prolog

Prolog est un langage extraordinaire, pas tant par ses possibilités effectives, mais parce qu'il nous montre qu'il peut exister d'autres moyens de programmer un ordinateur.

Prolog (PROgrammation LOGique) est né en France, à Marseille et a servi de base aux programmes de recherche japonais sur les ordinateurs de 5ème génération.

Ce qui est phénoménal, c'est qu'en Prolog, il nous suffit de décrire ce que l'on sait sur le domaine étudié pour constituer de ce que l'on appelle en Intelligence Artificielle une **base de connaissances**. Puis on décrit notre problème que Prolog se chargera de résoudre, sans qu'on n'ait à lui dire comment faire.

Il ne s'agira pas ici de détailler parfaitement tout le langage, mais de montrer progressivement les possibilités de Prolog, et donc des possibilités de programmation

qu'aucun langage classique n'autorise à ce jour. Des exemples choisis pour illustrer ces possibilités sont disponibles dans [l'Annexe 1](#) et [l'Annexe 2](#)

Avec Prolog, le « programme » se compose d'un ensemble de règles (sous la forme : *conclusion IF conditions*) et de faits (en général des affirmations : on précise ce qui est vrai, tout ce qui n'est pas précisé est faux ou inconnu).

Le langage gère des variables simples (entiers, réels,...) et des listes (mais pas les tableaux).

Les variables simples sont les entiers, les réels, les chaînes de caractères et les constantes symboliques (chaînes sans espace ni caractères spéciaux, commençant par une minuscule, mais ne nécessitant pas de *quotes*)

Une variable « contenant » une valeur est dite « liée » ou « instanciée », une variable de valeur inconnue est dite « libre ».

Prolog cherche à prouver que le but (*goal*) demandé est vrai. Pour cela, il analyse les règles, et considère comme faux tout ce qu'il n'a pas pu prouver. Quand le but contient une variable libre, Prolog la liera à toutes les possibilités

Une analyse de l'exemple proposé en [l'Annexe 1](#) permet de montrer que le comportement de Prolog est donc différent suivant que les variables soient liées (il cherche alors à prouver la véracité) ou libres (il essaie alors de les lier à des valeurs plausibles).

Ceci démontre une différence capitale entre un langage déclaratif (on dit simplement ce que l'on sait) et les langages procéduraux classiques (on explique à la machine comment résoudre le problème en prévoyant tous les cas par un algorithme).

Comme on l'a vu, en prolog, il n'y a pas possibilité d'utiliser de tableaux (par exemple pour accéder directement à la n^{ème} valeur). Par contre, il est possible de les remplacer par des listes. L'avantage des listes est que leur longueur est dynamique. Leur utilisation est récursive (on ne peut accéder à la n^{ème} qu'après avoir accédé aux n-1 précédentes).

Une liste constante est représentée entre crochets ([] signifiant ensemble vide) :

```
equipe(ulp,[jean,lucie,yves,anne,alain,julie,marc,luc]).
```

```
equipe(ushs,[jo,cathy,ahmed,louis,paul,eve,marie,rose]).
```

Lorsque l'on représente une liste par une variable, deux écritures sont possibles : soit par un nom de variable (commençant par une majuscule), soit par [Tete|Queue], où Tete (du type élément) est le premier de la liste, et Queue (de type liste d'éléments) le reste de la liste.

exemple :

```
afficher([T|Q]) :
```

```
writeln(T),nl,
```

```
afficher(Q).
```

```
afficher([]).
```

autre exemple, très utile :

```
appartient_à(X,[X|_]).
```

```
appartient_à(X,[_|Queue]) :- appartient_à(X,Queue).
```

Il n'est pas nécessaire de pousser plus en avant l'explication du code avec Prolog.

Ajoutons simplement que les listes peuvent servir à représenter des ensembles (les éléments appartiennent ou non à différents ensembles, que l'on peut combiner par l'union ou l'intersection), mais aussi être ordonnée (et entre autres pourront être triées pour faciliter l'analyse d'un résultat)

Il suffit d'analyser l'exemple proposé en [Annexe 2](#) qui démontre comment combiner les possibilités offertes par Prolog pour **obtenir ce que l'on cherche en utilisant un langage simple et convivial de règles, et sans forcément être obligé en tant que développeur de « penser à toutes les possibilités »...**

2. La 5ème génération de langage

2.1. Langage naturel et recherche d'informations

Une équipe de recherche de l'*Association Canadienne des Sciences de l'Information* a travaillé récemment sur un outil de génération de textes qui amène une **nouvelle visibilité sur les possibilités offertes par les langages de 5ème génération en matière de traitement de l'information.**

Ces travaux démontrent que l'expérience acquise en recherche d'information et en génération automatique de textes (production de textes par un système informatique à partir d'un contenu formel) montre que la relation de l'utilisateur avec le système de recherche d'information n'est pas naturelle, l'utilisateur devant généralement s'adapter au système.

Le processus de recherche d'information se compose de trois phases :

- x formulation de la requête par l'utilisateur,
- x traitement de la requête par le système,
- x affichage des résultats.

Quand la formulation de la requête est inadéquate, certains systèmes permettent d'en générer automatiquement différentes reformulations en fonction des caractéristiques du système de recherche d'information.

Il est également possible d'engendrer une explication sur la méthode employée et le type de données traitées par certains systèmes afin que l'utilisateur puisse comprendre le processus de recherche et adapter sa requête.

Une analyse précise de cette problématique est disponible en [Annexe 4](#)

Si une trop grande quantité d'informations est délivrée, il est nécessaire de structurer, d'organiser la présentation des résultats voire, selon l'application, de produire un résumé des réponses.

2.2. Les perspectives en recherche d'informations

L'évolution technologique des moteurs de recherche disponibles sur le marché ainsi que le nombre de partenariats actifs dans ce domaine montrent bien que si le problème n'est pas simple, la solution réside sans doute dans la combinaison des techniques disponibles.

Nous sommes donc encore loin du moteur de recherche parfaitement adapté à la pensée humaine. Mais nous nous approchons de celui adapté à la pensée machine... Le but à attendre est en cours d'identification, et les moyens sont ciblés : **il est nécessaire de définir des règles.**

2.3. Les technologies de règles

Les règles sont partout. On les trouve dans les langages de programmation, nous l'avons vu avec Prolog, mais aussi dans de nombreux domaines, disciplines et industries. Les politiques commerciales, les lois et les réglementations, les directives et recommandations, les définitions, les axiomes, les traductions de schémas de base de données, la répartition des flux de travail et contraintes techniques, tous ont besoin d'une approche déclarative et modulaire pour leur exécution.

Ces marchés en pleine expansion concernent plusieurs familles de technologies de règles : les règles de production, celles de type *événement-condition-action*, le langage Prolog, les systèmes de base de données relationnelles, et d'autres encore. **Même si l'interopérabilité entre ces systèmes est actuellement assez limitée, particulièrement entre ces différentes familles, on ne peut raisonnablement ignorer l'application de ces règles dans le Web.**

3.Lien entre la 5ème génération des langages et Web Sémantique

Les règles sont aussi un élément clé de la vision du Web sémantique

Elles permettent l'intégration, la dérivation, et la transformation de données provenant de sources multiples, et ce d'une façon distribuée, transparente et extensible. Les règles elles-mêmes peuvent être exploitées en tant que données, publiées sur le Web, et lorsque des URI (Uniform Resource Identifier) sont considérés comme des symboles dans un langage de règles, elles peuvent représenter des liens utiles entre les bases de connaissance. Par exemple dans le cadre des services Web (ou *Web Services*), les règles offrent la possibilité d'automatiser la mise en application et la composition de procédures régissant la distribution des informations, l'accès aux services, et l'exécution des opérations.

3.1.La sémantique, le message

Bien souvent la difficulté de compréhension d'un message ou d'une idée est liée à la façon dont elle est exprimée.

Le Web Sémantique et RDF (Resource Description Framework) sont partis du mauvais pied avec la première version du document RDF.

Il est certain que le document a pour but d'expliquer une technologie qui permet de définir un Web Sémantique alors tentons de l'analyser plus finement en analysant ce qu'en dit le W3C

(Une description plus précise des apports du W3C et de RDF sur le sujet sera faite ultérieurement, il ne s'agit ici que de mieux comprendre certains concepts)

L'introduction du document REF démarre ainsi :

«The Resource Description Framework (RDF) is a language for representing information about resources in the World Wide Web. It is particularly intended for representing metadata about Web resources, such as the title, author, and modification date of a Web page, copyright and licensing information about a Web document, or the availability schedule for some shared resource.»

Cela ne semble pas très clair, en effet cela n'explique pas vraiment la notion de Web Sémantique. Le sujet reste opaque.

Peut-être que la description de l'activité du Web Sémantique est plus claire et nous permettra de comprendre un peu mieux :

«Semantic comes from the Greek words for sign, signify, and significant, and today means of or relating to meaning, often in language. The Semantic Web is an extension of the current Web in which information is given well-defined meaning, better enabling computers and people to work in cooperation. The mix of content on the Web has been shifting from exclusively human-oriented content to more and more data content. The Semantic Web brings to the Web the idea of having data defined and linked in a way that it can be used for more effective discovery, automation, integration, and reuse across various applications.»

Que faire pour trouver une signification à la portée de tous ?

Voyons ce que dit Google, le plus utilisé des moteurs de recherche à l'heure actuelle sur Internet, pour trouver une signification à la portée de tous. La requête de recherche se limitera ici au mot : « Sémantique »

Je fini par trouver, au milieu de centaine de réponses inutiles, ou plutôt hors du sujet..., l'extrait d'un livre (*Semantics* par F. H. George du Département de Psychologie, Université de Bristol)

Voilà ce que dit l'introduction du livre :

« This book is about semantics, and we shall define semantics, rather generally, as the science of meaning. Such a definition is meant to imply that we can regard semantics as being directly concerned with anything and everything that can be classed as either signs or symbols. These signs and symbols may be road-signs, clouds or gestures on one hand, or words and sentences on the other. They can, in other words, be iether linguistic or non linguistic, and yet are the subject of semantic study wherever they are used to carry information.

This process of carrying information may involve, and generally will involve, people. Ths we are very especially concerned with the communication that exists between person and person. In spite of this we may also be involved with machines, the inanimate world in which we live, and indeed anything at all which is either a source or destination of meaningful communication.

Semantics is not an easy subject to write about clearly, precisely because semantics is mainly concerned with writing and talking; and writing, about, or talking about, talking can become very confusing, unless we are especially careful and especially clear. [...] »

Un peu plus loin vient l'argument principal du 1er chapitre :

« SEMANTICS is the study of meaning, mainly in language. Languages are like maps of territory, they may depict relationships accurately or they may not. Some relationships are very difficult to map because of the inherent complexity of those relationships, or because they are spread out over time as well as space.

The idea that languages are maps is not literally true, it is a metaphor to indicate the fact that languages describe states of affairs, more or less, accurately.

In language, we may state truth or falsehood, valid arguments or invalid arguments, and the combination of these two is the basis of all human knowledge. We might even argue that what cannot be put into words does not constitute knowledge, although this may be regarded as a rather extreme view. Certainly all of scientific knowledge depends on your ability to state that knowledge in words. Semantics and science are very closely bound up with the same aim of trying to make our ideas and our knowledge clear and available to other people. »

En relisant à nouveau ces passages et en gardant en mémoire la notion de *Web* et de *Documents en ligne sur un réseau*. Le concept de Web Sémantique devient beaucoup plus clair.

3.2. De la mémoire collective vers le Web Sémantique

Il est très difficile de résumer ici les nombreuses évolutions touchant le Web, les technologies de l'information et de la mémoire, les outils et méthodes de recherche d'information.

Les évolutions du côté de la structuration et de l'indexation de l'information représente les mutations les plus profondes autour d'Internet. Elles sont conditionnées par de très nombreuses recherches autour du **Web Sémantique**, d' **XML**, des **méta données**, de la **normalisation** et de l'essor des **techniques d'indexation**.

Les réponses à quelques questions qui se posent de nouveau ont été trouvées sur différents sites internet. Tous liés de près ou de loin à celui du W3C ([c.f. References](#)):

Qu'est-ce que le W3C ?

Le W3C (World Wide Web Consortium) a été créé en 1994 par Tim Berners-Lee et le MIT (Massachusetts Institute of Technology), pour gérer les technologies et les évolutions du Web.

Ce consortium international, dirigé par T. Berners-Lee, repose sur trois grands organismes de recherche, qui sont les hôtes du W3C : le MIT à Cambridge, la Keio University au Japon, l'INRIA en France.

Outre ces trois organismes, le W3C est composé de nombreux membres :

- x grandes entreprises d'informatique : Adobe Systems, Apple, Bull, IBM, Cisco Systems, Compaq, Intel, Microsoft, Xerox.
- x opérateurs de télécommunications : AT&T, France Télécom, Deutsche Telekom, et d'électricité : EDF
- x grandes firmes : Alcatel, Matra Hachette, AOL, Boeing
- x autres laboratoires et organismes de recherche : CNRS, le CERN (Centre d'Etudes et de Recherche Nucléaire), le laboratoire européen de physique des particules à Genève, qui est le berceau du WWW
- x institutions militaires : l'OTAN est membre du W3C

Les activités du W3C concernent :

- x l'architecture du Web : serveurs Web, protocole HTTP...
- x l'évolution du langage HTML

- x les interfaces utilisateurs
- x les spécifications XML

Le W3C regroupe actuellement plus de 510 membres. Il ne produit pas des normes, mais des « Recommandations », diffusées gratuitement.

Qu'est-ce que le Web Sémantique ?

A ce stade il s'agit d'un vaste chantier de recherche, mobilisant de nombreux acteurs, lancé et piloté par le W3C pour pallier les insuffisances actuelles du Web.

Quels sont les finalités et les objectifs du Web Sémantique ?

Les limites et les inconvénients du Web actuel sont l'hétérogénéité des formats, des informations, l'absence de description et d'indexation des ressources, l'imprécision de la recherche d'information.

En résumé le problème provient de l'absence de structure explicite globale du Web : Un vaste réseau de noeuds et de liens, mais pas d'exploitation sémantique des liens hypertexte

Ainsi le but avoué des recherches autour du Web Sémantique sont bien de transformer le Web en un vaste « *espace d'échanges de ressources entre machines, permettant l'exploitation de grands volumes d'informations et de services* » ainsi que de « *libérer les utilisateurs d'une partie du travail de recherche et d'exploitation des résultats, grâce à des capacités accrues de recherche d'information, d'intégration de sources d'information, de découverte, d'exploitation et de combinaisons de services et de raisonnement des machines* » (d'après Laublet)

Par conséquent les objectifs visés sont :

- x une meilleure structuration du Web
- x rendre explicites les relations sémantiques (les liens) entre les documents du Web
- x faciliter l'utilisation et la recombinaison des ressources par les machines
- x ajouter des annotations sémantiques aux ressources du Web, décrivant leurs contenus et leurs fonctionnalités
- x permettre une meilleure interopérabilité : des ressources et des machines
- x développer une grammaire universelle pour la production, le stockage et l'échange des données (comme XML)

L'enjeu qui sort de toutes ces objectifs : **permettre et développer un accès « intelligent » à l'information**

Sur quels principes et outils repose le Web Sémantique ?

Pour permettre aux machines d'exploiter ces annotations sémantiques et permettre ces accès intelligents aux ressources, quatre normalisations sont nécessaires :

- x normaliser l'identification des ressources numériques
- x normaliser la description des ressources : les systèmes de méta données (RDF, Dublin Core...)
- x normaliser la structuration des documents numériques, c'est à dire créer une langue universelle pour les documents numériques : XML
- x normaliser l'indexation, c'est à dire les langages permettant de décrire et d'indexer le contenu des documents : classifications, ontologies, thésaurus...

Principe fondamental du Web Sémantique : **la séparation du contenu des documents de l'organisation de ce contenu**

Quels sont les grands travaux de ce chantier de recherche ?

Les travaux en cours semblent être menés dans quatre directions distinctes :

- x l'identification stable des concepts et des objets du Web : travaux menés sur les normes d'identification des documents électroniques
- x le métalangage XML et la normalisation des modèles de structuration des documents
- x les outils et les normes d'indexation des documents : les méta données
- x la construction d'ontologies et de thésaurus partageables pour l'indexation

L'Ontologie, qu'est ce que c'est ? :

A l'origine, il s'agit d'un terme philosophique (la science de l'être). Le terme a été repris en informatique pour désigner la représentation des connaissances et la définition de catégories. Une ontologie structure les termes d'un domaine, en établissant des relations de proximité entre eux, du type « partie de ». Les ontologies informatiques sont des formes de classifications

Quelles peuvent être les conséquences du Web Sémantique ?

A moyen terme, si le Web Sémantique se développe véritablement et s'étend à la plupart des ressources numériques du Web, de profonds bouleversements sont à prévoir dans la production, l'échange et la recherche d'informations sur le Web :

- x travail en profondeur dans la trame même des documents et de l'information, au niveau « micro » des documents ;
- x importance de la notion de « granularité » de l'information
- x possibilités inédites de recherche intelligente sur le contenu
- x nouvelles formes de représentation de l'information : en amont (lors de la conception avec XML) et en aval (lors de la recherche)

3.3. De la sémantique des textes au Web Sémantique

Comme on l'a déjà vu depuis une dizaine d'années, chercheurs et enseignants du monde entier se mobilisent dans un large mouvement pour le libre accès aux textes et aux textes de la recherche scientifique en particulier.

Steven Harnad remarquait en 1991 que l'écriture électronique était, après le langage, l'écriture et l'imprimerie, « *la quatrième révolution cognitive de l'humanité.* »

Depuis l'avènement de l'Internet notre société moderne subit une mutation majeure que les scientifiques sont les premiers à reconnaître en signalant les nouvelles dimensions qui s'offrent à la pensée humaine. C'est cette transformation radicale du cycle des connaissances scientifiques qui est à la base de la Déclaration de Berlin, dont le préambule énonce la relation entre les possibilités technologiques de l'Internet et la constitution d'une « représentation globale et interactive de la connaissance humaine » :

L'Internet a fondamentalement transformé les réalités matérielles et économiques de

la diffusion de la connaissance scientifique et du patrimoine culturel. Pour la toute première fois, l'Internet nous offre la possibilité de constituer une représentation globale et interactive de la connaissance humaine, y compris son patrimoine culturel, et la garantie d'un accès mondial.

« Conformément à l'esprit de la Déclaration de Budapest pour l'accès ouvert, de la charte ECHO et de la Déclaration de Bethesda pour l'édition en libre accès, nous avons élaboré la déclaration de Berlin pour promouvoir un Internet qui soit un instrument fonctionnel au service d'une base de connaissance globale et de la pensée humaine, et de définir des mesures qui sont à envisager par les responsables politiques en charge de la science, les institutions de recherche, les agences de financement, les bibliothèques, les archives et les musées. »

(Déclaration de Berlin,Préface)

Il s'agit là des début de constitution d'une « **base de connaissance globale** »

On sait que la publication scientifique diffère radicalement de toute autre publication par sa vocation : le chercheur ne cherche pas de bénéfices financiers, mais une reconnaissance auprès de ses pairs qui accèdent et valident son travail intellectuel. Un chercheur qui ne publie pas signe son arrêt de mort scientifique. Mais pour être validée la publication scientifique doit d'abord être visible.

Bien que l'édition électronique ait facilité l'accès à la publication en général, elle n'a pas résolu, pour autant, les problèmes de visibilité

On peut avoir des milliers de pages sur un thème quelconque quelque part sur Internet, et ne pas se douter de leur existence.

Il est évident qu'aucune archive scientifique ne peut exister sur Internet sans un minimum de méta données, et qui plus est, sans une convention communément admise sur la description et l'encodage des méta données.

L'unification des méta données mène inévitablement au problème des standards. Les standards sont, évidemment, des systèmes de restrictions

Tout d'abord, il est nécessaire de rappeler que l'Internet est devenu possible notamment grâce à des ensembles de standards communs appelés protocoles :

- x D'abord par un standard de compatibilité entre ordinateurs de différentes marques ce qui permet leur interconnectivité et la création des réseaux.
- x La véritable explosion de ces réseaux s'opère à partir de 1982 grâce à l'adoption d'un protocole commun des transferts, le TCP/IP (*transmission contrôle protocol/internet protocol*).
- x Le pas décisif est l'adoption en 1992 du protocole HTTP (*Hyper Text Transfer Protocol*) et du langage HTML (*HyperText Markup Language*) comme des langages standards internationaux d'encodage de fichiers et de navigation à travers les réseaux à l'aide de liens hypertextuels. C'est le véritable début d'Internet.

La question des standards n'est donc pas négligeable puisque c'est l'implémentation des standards à l'échelle internationale qui a donné toute leur vitalité à l'Internet et au WEB ; qu'on pense à l'exemple du Minitel qui n'a pas réalisé les développements technologiques nécessaires pour s'adapter.

La question des méta données est donc particulièrement importante pour un Web classique dit « syntaxique », et encore plus pour un Web Sémantique où les méta données sont au coeur de la recommandation RDF du W3C.

Concernant le projet de Web Sémantique, Picouet et Saglio expliquent que « *En ajoutant des méta données aux documents, on souhaite rajouter au Web la sémantique qui lui manquait* »

Mais attention, la méta donnée (méta-donnée = donnée sur la donnée) n'est qu'une formalisation d'un contenu sémantique relatif à la donnée elle-même. Autrement dit, la donnée possède une sémantique préalable à toute méta donnée, sans quoi aucune formalisation ne serait possible. **La méta donnée ne rajoute pas une sémantique, elle la décrit.**

Il me semble donc à ce stade que le Web Sémantique de demain ne sera pas « plus sémantique » qu'aujourd'hui, il sera tout simplement mieux interprétable par les machines, et par conséquent mieux catalogué et répertorié pour l'homme.

Bien qu'il existe actuellement des conventions sur les méta données comme le schéma Dublin Core, un consensus général est attendu et l'élaboration des standards

de méta données est ouverte aux contributions. Dans cette perspective, il faudrait envisager la question des méta données non seulement dans son aspect technique (indexation), mais aussi dans ses bases théoriques, notamment par un raisonnement sur le socle général des méta données sémantiques et les ontologies.

Par un autre aspect du traitement des textes sur Internet, il apparaît que les flottements dans l'évolution du Web poussent à mener la réflexion éditoriale au-delà du cadre restreint d'une seule publication scientifique : **les projets de création d'archives ouvertes sont un dépassement des limites de l'édition traditionnelle, il faut donc se poser la question de l'éthique**

La création d'archives de la recherche accessibles au monde entier est un grand projet de l'avenir non seulement par son ampleur, mais surtout par sa valeur éthique et scientifique. L'accessibilité des archives médicales, par exemple, profitera à tous les pays défavorisés qui n'ont pas les moyens d'investir dans la recherche. Sans parler là d'un grand projet humaniste, les enjeux sont donc importants et il serait prudent d'évaluer les leçons du passé.

Suivant les derniers sondages et contrairement aux idées reçues la volonté des européens de contribuer aux archives ouvertes est aujourd'hui en hausse :

	oui	non
Pays européens (EU)	31%	69%
Pays européens (hors EU)	31%	69%
UK	27%	73%
US	27%	73%
Canada	18%	82%

Taux d'acceptabilité de l'archivage parmi les chercheurs qui n'ont jamais publié en libre accès
(Swan & Brown 2005)

L'avantage des archives spécialisées est qu'elles attirent un public expert : la plupart des chercheurs s'intéressent à des domaines spécifiques et préfèrent avoir accès à des collections restreintes mais avec une plus grande concentration de contenus pertinents dans le domaine-cible, plutôt que de devoir faire des recherches dans des

archives vastes et hétérogènes.

Par conséquent il y a une forte motivation à collaborer aux archives thématiques. Les chercheurs en sciences humaines déclarent préférer publier dans des sites Web. Il s'agit donc d'en assouplir les démarches.

En guise de conclusion, la question de l'homogénéité des archives mène naturellement à des problématiques : au sein du projet de Web Sémantique une nouvelle solution s'impose depuis quelques années, appelé « Web communautaire ».

L'idée est d'accélérer la création du Web Sémantique par un puzzle de Webs locaux et régionaux. Ainsi l'on a vu naître des projets de « Webs Sémantiques locaux et intranets sémantiques », « Web médical », « Web muséal » (ex: projet européen MESMUSES), « Web Sémantique d'entreprise » (WSE), etc. (Plus d'informations dans le dossier des rapports de recherche « Web Sémantique » de l'Association française d'Intelligence Artificielle.

Bien qu'actuellement la recherche sur les Web locaux et l'élaboration des archives spécialisées restent deux pôles d'action bien étanches, la similitude entre les deux démarches est évidente : les deux projets cherchent à asseoir leurs objectifs sur les compétences des personnes et des institutions expertes dans un domaine donné.

Ainsi il est clair que ***toute démarche sur le Web Sémantique doit commencer par une adaptation des contenus, car c'est là que se trouvent les vrais objets à décrire, les données .***

Une archive ouverte ou un Web Sémantique ne fait que pourvoir ces données d'un jeu de méta données pour les rendre interopérables dans le cadre d'un environnement spécifique.

4. Le Web Sémantique

Attention, comme on l'a déjà vu le Web Sémantique est en aucun cas un langage de programmation a proprement parlé. Il s'agit plutôt d'un projet qui tend à créer un moyen universel d'échange d'informations, en donnant du sens aux contenus existants. Pour que ces contenus soient compris non simplement par les utilisateurs, mais aussi par les machines. Il se compose d'un ensemble de spécifications, de langages et d'outils.

4.1. La spécification RDF

C'est au cours de la dixième édition de la World Wide Web Conference où il a beaucoup été question de Web Sémantique que s'est confirmée la controverse dont la spécification RDF fait l'objet.

Aujourd'hui, comme nous l'avons déjà vu, nous naviguons sur un Web simple, composé de pages au format HTML, transitant sur le protocole HTTP, et comme expliqué auparavant le nombre de pages est devenu si important qu'il est bien difficile de ne pas se perdre dans ce gisement d'informations.

Pour interpréter les contenus du Web, nous parvenons à un stade où les machines doivent impérativement être mieux utilisées.

Cette interprétation des ressources du Web par des machines implique trois pré requis :

- x un codage universel des caractères : en l'occurrence Unicode
- x une structure standard de documents ; XML est bien parti pour répondre à ce besoin
- x une définition du sens des documents.

Le développement de ce troisième point conduit au Web Sémantique.

D'où provient RDF ?

Ce concept a-t-il un rapport avec la notion de méta données ?

Et s'appuie-t-il sur les standards XML recommandés par le W3C, comme XML Schema ?

Pour matérialiser cela, et plutôt que XML Schema, **Tim Berners-Lee** (W3C) présente un langage conforme au formalisme XML, baptisé RDF (Resource Description Framework).

Son but est d'exprimer des faits comme par exemple, « le ciel est bleu ».

RDF fournit les méta données d'un document.

Ce langage s'appuie sur des « triplets », formés de l'identifiant de la ressource décrite (Resource), de sa propriété (Property) et de sa valeur (Value).

Un identifiant URI (Uniform Resource Identifier) sera donné à chaque page du Web, mais aussi à tout élément inclus dans un document XML.

Dans l'exemple « le ciel est bleu », « ciel », « est », le verbe être, et « bleu », la couleur, seront chacun associés à un URI.

Le W3C s'intéresse au RDF depuis quelques temps.

L'activité « Semantic Web » a été lancée en février 2001. Dans ce cadre, le W3C alloue des ressources pour donner suite aux travaux déjà menés, cette activité succédant à celle consacrée aux métadonnées. Un nouveau groupe de travail, *RDF Core*, a été créé à cette occasion. Il reprendra en particulier les recherches effectuées par le précédent groupe, *RDF Interest*.

Le consortium du Web n'est pas le seul à marquer son intérêt pour cette notion. La Commission européenne vient, en effet, d'annoncer qu'elle publiera en juin 2001 une nouvelle ligne d'action en rapport avec le Web Sémantique, fondée sur les conclusions de l'atelier *Semantic Web Technologies Workshop*.

Les réflexions porteront sur la structure du contenu numérique et la définition de son sens, l'extraction des attributs sémantiques d'un contenu, la recherche et le filtrage des connaissances, ainsi que sur le développement d'interfaces visuelles intelligentes faisant appel à des structures sémantiques.

Cette initiative de l'Union européenne n'a d'autre but que de faire prendre conscience aux industriels européens des besoins engendrés par cette nouvelle étape du Web.

L'acceptation de RDF

Si ce langage a bénéficié d'un bon accueil dans le milieu universitaire, il ne semble pas emporter les faveurs de l'industrie. RDF fait l'objet d'une sérieuse controverse.

Les experts des systèmes de gestion de la connaissance jugent d'abord que la problématique de leur domaine est trop ardue pour être résolue par une proposition aussi simple.

RDF est aussi discuté par les industriels, qui préfèrent axer leurs efforts sur un secteur qu'ils jugent plus utile, à savoir les échanges collaboratifs du commerce électronique.

On se retrouve donc actuellement devant deux types de Web Sémantique

- x le premier lié aux contenus documentaires
- x le second aux services de commerce électronique

Ces utilisations doivent donc être convenablement identifiées.

Et c'est ce que la communauté tente de faire avec les récentes technologies XML que sont Soap, le protocole XML, WSDL (le standard de description de ces services) et UDDI (l'annuaire universel qui stocke leur référence).

Certaines applications de RDF laissent cependant supposer que celui-ci va se répandre.

Le W3C a ainsi lancé l'application *Annotea*, qui repose sur RDF et permet aux lecteurs d'une page de rédiger leurs propres annotations.

Dublin Core, qui commence à faire référence, permet de définir un mécanisme de classification de documents. Il en existe deux versions :

- x l'une fondée sur les balises « Méta » de HTML
- x l'autre sur le langage RSS (RDF Site Summary) constitue une autre technologie largement utilisée par des sites portails et les Weblogs pour la propagation de titres. Ce vocabulaire RDF a été inventé par Netscape pour syndiquer les titres des informations publiées sur leur portail.

En résumé, RDF est un standard permettant la mise en place de descriptions simples. XML est à la syntaxe, ce que RDF est à la sémantique. RDF Schema permet ensuite de combiner ces descriptions en un seul vocabulaire. A tout ceci, il manque la possibilité de décrire des vocabulaires spécifiques à des domaines bien particuliers. C'est là que les ontologies jouent leur rôle.

On peut donc considérer que l'état actuel du marché démontre que le Web Sémantique verra le jour, avec ou sans RDF, mais pas sans le W3C...

4.2. Les actions du W3C autour d'un langage

Le Consortium World Wide Web (W3C : <http://www.w3.org/>.) a été créé pour mener le Web à son plein potentiel en développant des protocoles communs qui facilitent son évolution et assurent son interopérabilité. C'est un consortium industriel international, piloté conjointement par l'European Research Consortium for Informatics and Mathematics (ERCIM) basé en France, l'Université de Keio au Japon, et le MIT Laboratory for Computer Science (MIT LCS) aux Etats-Unis.

Les services fournis par le Consortium se composent de :

- x la constitution et la mise à disposition d'informations concernant le World Wide Web à destination des développeurs et des utilisateurs.
- x la mise en oeuvre de logiciels permettant d'incorporer et de promouvoir les standards
- x la mise en place de diverses applications prototypes visant à démontrer l'utilisation des nouvelles technologies.

Aujourd'hui, le Consortium compte près de 400 membres.

Le W3C publie le Langage d'Ontologies Web en *Recommandation Candidate* (cf. [L'Ontologie qu'est ce que c'est ?](#)).

Je souhaite simplement préciser qu'une ontologie définit les termes utilisés pour décrire et représenter un champ d'expertise. Les ontologies sont utilisées par les personnes, les bases de données, et les applications qui ont besoin de partager des informations relatives à un domaine bien spécifique, comme la médecine, la fabrication d'outils, l'immobilier, la réparation d'automobiles, la gestion de finances, etc. Les ontologies associent les concepts de base d'un domaine précis et les relations entre ces concepts, tout cela d'une manière compréhensible par les machines. Elles encodent la connaissance d'un domaine particulier ainsi que les connaissances qui recouvrent d'autres domaines, ce qui permet de rendre les connaissances réutilisables.

Et c'est via OWL que vont se renforcer les fondations du Web Sémantique pour le W3C.

Le mode de fonctionnement général du W3C fait que l'avancement d'un document au stade de *Recommandation Candidate W3C* constitue un appel à des implémentations de la spécification. Cela indique que le document a été révisé par les autres groupes de travail W3C, et assure que la spécification est stable.

OWL est un langage permettant de définir des ontologies Web structurées, et ainsi autorise une intégration plus riche et garantit l'interopérabilité des données au travers d'applications.

Comme on l'a déjà vu, la bioinformatique, le secteur de la santé, les entreprises corporatives et les gouvernements représentent les premiers utilisateurs de ce standard car OWL permet de créer un éventail large d'applications allant des portails Web, de la gestion des bibliothèques, de la recherche de contenu, de mise en place d'agents intelligents, jusqu'aux Services Web.

« OWL est une étape importante pour que les données sur le Web soient plus facilement interprétables par les machines et réutilisables au travers de nombreuses applications. Il est encourageant de voir qu'OWL est déjà utilisé comme un standard ouvert pour le déploiement d'ontologies à grande échelle sur le Web. »

explique Tim Berners-Lee, Directeur du W3C.

Le Groupe de Travail *Web Ontology* du W3C a produit six documents OWL. Chacun d'entre eux adresse un segment précis relatif à la spécification, ce qui permet un apprentissage, une compréhension et une utilisation du langage OWL simplifiés. Les six documents sont :

- x un aperçu d'OWL (présente brièvement les caractéristiques d'OWL et la manière de les utiliser)
- x une syntaxe abstraite et une sémantique d'OWL (les détails du mapping d'OWL à RDF)
- x des conditions préalables et des cas d'utilisation d'OWL (qui ont motivé les travaux sur OWL)

- x des exemples de tests d'OWL (fournissant plus d'une centaine de tests pour s'assurer que les implémentations OWL sont cohérentes avec la conception du langage)
- x un guide OWL (il est compréhensif et illustré d'exemples d'utilisation des caractéristiques d'OWL)
- x et une référence OWL (qui fournit les détails de chaque caractéristique d'OWL)

Une liste de questions fréquemment posées (FAQ) est disponible pour plus de détails sur la spécification OWL, et il est aisé d'imaginer qu'il s'agit là de la principale source d'information pour la rédaction de cette thèse professionnelle...

OWL s'appuie sur un modèle et un schéma RDF pour ajouter plus de vocabulaire dans la description de propriétés et de classes. On peut donc se demander ce qu'apporte OWL.

En effet il s'agit d' un langage d'ontologies Web. Mais les premiers langages utilisés pour le développement d'outils pour des communautés d'utilisateurs spécifiques (particulièrement en Sciences et dans des applications d'e-commerce spécifiques à certaines entreprises) n'ont pas été définis pour être compatibles avec l'architecture du Web en général, et du Web Sémantique en particulier.

OWL répare ce manque en utilisant à la fois les URIs pour nommer et la fonctionnalité fournie par RDF ([*cf.4.1.La spécification RDF*](#)) pour créer des liens.

Ainsi, les ontologies Web possèdent les avantages suivants :

- x Capacité d'être distribué au travers de nombreux systèmes
- x Mise à échelle pour les besoins du Web
- x Compatible avec les standards Web pour l'accessibilité et l'internationalisation
- x Ouvert et extensible

En résumé, L'activité Web Sémantique du W3C s'appuie sur les travaux réalisés au sein de l'activité XML principalement, et son objectif est de mettre en place des technologies standards fondées sur XML, et qui soutiennent le développement du Web Sémantique.

4.3.Des exemples et des applications

4.3.1 Le laboratoire de Web Sémantique (LabWebSem)

Le Laboratoire de Web Sémantique de l'Institut de technologie de l'information du CNRC (ITI-CNRC) a été mis sur pied en 2002 pour élaborer des outils et des applications de Web Sémantique, de même que pour coordonner des travaux semblables réalisés au Canada et partout dans le monde.

Le LabWebSem élabore des ontologies composées de taxonomies qui classent les objets Web d'après des règles et qui les typent par taxonomies, pour assurer l'intégrité et l'inférence de connaissances. Le LabWebSem étudie également les agents qui font appel aux ontologies notamment pour récupérer des similitudes et composer des objets d'apprentissage.

Le LabWebSem met l'accent sur l'extraction des méta données pour traiter le très grand nombre d'objets Web qui constituent des documents en langage naturel.

Le LabWebSem a lancé une initiative qui fait appel aux techniques du Web Sémantique dans le but d'édifier un *Nouveau-Brunswick sémantique* (Province canadienne),.c'est à dire l'utilisation des techniques du Web Sémantique pour stimuler le commerce électronique. Les participants peuvent ainsi se faire connaître, consulter des catalogues de produits et de services, de même que des lignes directrices et des règles contractuelles, ainsi que trouver des occasions d'affaires et des règlements gouvernementaux. L'« interface sémantique » commune du Nouveau-Brunswick étant utilisable indifféremment par des entreprises et des organismes gouvernementaux ainsi que des clients et des citoyens.

Le LabWebSem participe à toutes sortes de projets et collabore avec divers organismes de normalisation comme le Consortium W3C, OASIS, OMG et RuleML.

Comme on l'a vu, le Web Sémantique nécessite une normalisation à tous les niveaux, des langages et des outils ontologiques aux plates-formes complètes de commerce électronique. Le LabWebSem sert également de dépôt pour les logiciels du Web Sémantique au CNRC.

Voici un extrait des prototypes du LabWebSem, accompagnés de liens vers les

descriptions et les chercheurs.

RuleML : Le langage RuleML (**R**ule **M**arkup **L**anguage) vise à établir une interlangue du Web « classique » pour le traitement des règles administratives, et ce à l'aide du langage XML et de la sémantique formelle, et d'après des procédures de mises en œuvre efficaces. Le langage RuleML vient d'ailleurs d'intégrer la modélisation orienté objet et a été redéfini dans un schéma XML.

Metaxtract : Le projet Metaxtract permet l'annotation sémantique automatique ou semi automatique utilisant des techniques linguistiques sur des ressources riches en textes.

4.3.2 Triple

Triple est un langage de règles pour le Web Sémantique et son utilisation dans l'apprentissage électronique, ou e-learning.

Il a été spécialement conçu pour l'interrogation et la transformation de modèles RDF. Le système de modules de Triple permet de définir facilement des caractéristiques orientées objet pour différents modèles orientés objet et d'autres modèles de données comme UML, Topic Maps (schémas heuristiques) ou RDF Schema.

4.3.3 Dans l'entreprise

D'un point de vue opérationnel, le Web Sémantique fourni un cadre de travail pour la gestion des ressources, l'intégration, le partage et la réutilisation des données sur le Web. Ces formats de partage de données se rapportent aux applications, à la vie des entreprises et à celle d'autres communautés. Avec ces deux nouveaux standards, tous ces différents types "d'utilisateurs" peuvent désormais partager les mêmes informations, même s'ils ne partagent pas les mêmes logiciels.

Le grand intérêt des serveurs de connaissances réside dans leur capacité à intégrer l'existant. De plus, c'est un moyen simple d'adresser la gestion des connaissances explicites en relation avec les usages et processus métiers. Libérée d'une importante partie de cette tâche, l'entreprise peut, de ce fait, s'employer à réfléchir à la meilleure façon de traiter les connaissances tacites. Au final, le web sémantique peut

jouer un rôle central dans le cadre du *knowledge management*.

Même si le Web Sémantique semble promis à un bel avenir en entreprise, il faut noter que cette approche ne présente pas, à l'heure actuelle, que des avantages.

En effet, les entreprises qui souhaitent se lancer dans la mise en place d'ontologies sur l'ensemble de leur système d'information risquent de se trouver confrontées à quelques difficultés, essentiellement dues à la relative « jeunesse » du marché.

D'une part, les outils permettant de créer et de gérer les ontologies ne sont pas encore assez matures pour permettre « l'industrialisation » de l'utilisation des ontologies dans les entreprises. Ils ne sont pas d'accès faciles et il n'existe pas encore de véritables standards.

Toutefois, certains éditeurs de logiciels, comme *Mondeca* (<http://www.mondeca.com/fr/>), proposent des solutions complètes qui intègrent les différents standards via une approche innovante fondée sur l'utilisation d'Ontologie, de Sémantique et de représentation des connaissances au sein d'un même outil. La superposition de ces trois dimensions permet d'embrasser l'ensemble de la chaîne de traitement de l'information et des connaissances : indexation, modélisation, navigation et recherche, diffusion multi canal.

Actuellement, il n'existe pas de méthodologies prouvées et surtout éprouvées pour guider les entreprises dans la création d'ontologies. La création d'une ontologie est une affaire de spécialiste dans la mesure où il faut avoir des compétences à la fois dans le domaine à modéliser mais aussi en représentation des connaissances et linguistiques. Une pluridisciplinarité encore rare...

On peut donc considérer que les organisations qui entreprennent dès aujourd'hui la mise en place de serveurs de connaissances basés sur l'infrastructure du Web Sémantique posséderont une avance certaine quant à l'exploitation de leurs connaissances, documents, rapports et données en général.

Conclusion

Les nouvelles orientations prises dans le développement et l'utilisation des langages de dernière génération permettent aux développeurs de se concentrer sur les buts qu'ils doivent atteindre, plus que sur la manière de les atteindre.

Il apparaît nettement qu'on ne peut encore attribuer sans y réfléchir la dénomination de 5ème génération à ces langages émergents qui pourtant, contrairement à d'autres langages souvent lents et complexes, apportent des perspectives motivantes. Grâce en particulier à un niveau d'abstraction très élevé.

Le code « dit » en L5G est beaucoup plus concis. Il est plus rapide et plus fiable. Il ouvre des perspectives sur l'analyse sémantique des contenus plus que sur la simple logique des algorithmes de résolution de problématiques.

En cela je constate que ce souci d'**instauration de règles**, de simplification, de ré-utilisation (de ne pas « ré-inventer la roue ») a directement conduit dans le monde de l'Internet à ce que soient posées les bonnes questions. Et ce dans un souci perpétuel d'adaptation des technologies au monde existant.

La notion de Web Sémantique a déjà fait couler beaucoup d'encre, et il a été notamment écrit que cette technologie devait remplacer le Web que nous connaissons aujourd'hui.

Il me semble plutôt que le Web Sémantique se crée de manière incrémentale, en ajoutant pas à pas des descriptions aux données et documents déjà existants sur le Web. Ensemble, ces descriptions et les différentes manières de les connecter, les comparer et les opposer, rendent possible la mise en place d'applications, d'outils, de moteurs de recherche, d'agents, et tout cela sans changement apparent des pages Web.

Tout les métiers de traitement de l'information et de développements légers voient s'ouvrir des perspectives immenses (peut-être même infinies ...).

Il ne s'agirait pas ici de trop anticiper, mais le déclin des développements de systèmes lourds se présenterait-il ? Puisque tout existe quelque part dans nos systèmes, et que la sémantique peut nous aider à trouver ce que l'on recherche.

Une petite touche d'humanité encore, et plus particulièrement de sensibilité féminine pour terminer cette étude. Je me permet d'inciter à la lecture attentive de cette interview (cf. [Annexe 5](#)) réalisée par les membres de l'association *Internénettes* (<http://www.internenettes.fr>), et qui permet de résumer et mieux comprendre ce qu'est finalement le Web Sémantique.

Et je me dois d'avouer que c'est ce document qui avant tout autre m'a indéniablement poussé à travailler sur ce sujet... alors merci mesdames !

Références

- W3C : <http://www.w3.org/>
- *De la sémantique des textes au Web Sémantique.*
KYHENG, Rossitza. juin 2005, vol. X
- URFIST Bretagne-Pays de Loire, Alexandre Serres, 2002
- Université Sorbonne - Recherche Web Sémantique
<http://www.lalic.paris4.sorbonne.fr/stic/resume5.html>
- P. Trau - IPST-ULP - Janvier 2002.
- Sourceforge
<http://sourceforge.net/>
- Dmoz
<http://dmoz.org/Computers/Programming/Languages/> Annuaire mondial de recherche.
- Wikipedia
<http://en.wikipedia.org/wiki/Programmation/>
Plus de détails sur les principaux langages
http://en.wikipedia.org/wiki/Semantic_Web
Plus de détails sur le web sémantique
- Haton J.-P. et M.-C. (1993), *L'intelligence artificielle*, 3ème édition
Presses Universitaires de France.
- BRUNET, Étienne. *Les liens hypertextuels ou abondance de liens ne nuit pas*,
Lexicometrica, 1997
- Balicco L. (1997), « *Génération automatique de descriptions textuelles de figures à deux dimensions* »,
- Fluhr C. (1992), « *Le traitement du langage naturel dans la recherche d'information documentaire* »
- RASTIER, François. *De la signification au sens. Pour une sémiotique sans ontologie.* 2003
- http://solutions.journaldunet.com/0310/031003_tribune.shtml
Le web sémantique au secours des systèmes de knowledge management
- <http://www.mondeca.com/fr/>
Editeur spécialisé dans la gestion des taxonomies, référentiel métier, représentation sémantique des connaissances

Bibliographie chronologique

Les années 1960

(1964) Semantics par F. H. George (Département de Psychologie, Université de Bristol), The English Universities Press LTD

Les années 1970

(1973) MERTON, Robert K. The Sociology of Science: Theoretical and Empirical Investigations. Edited by Norman W. Storer. Chicago: University of Chicago Press, Ch. 13 : The Normative Structure of Science , p. 267-278.

(1975) BOURDIEU, Pierre . La spécificité du champ scientifique et les conditions sociales du progrès de la raison. Sociologie et sociétés, vol. 7, n° 1, mai 1975, p. 91-118.

Les années 1980

(1989) Dale R. , Generating referring expressions in a domain of objects and processes, PhD Thesis, Edinburg, Ecosse.

(1989) Danlos L. , « Génération automatique de textes en langue naturelle », Ann. Télécommun. 44, n° 1-2, France.

(1989) RASTIER, François. Sens et textualité. Paris : Hachette.

(1989) Miège B. , La société conquise par la communication, Grenoble : Presses Universitaires de Grenoble.

Les années 1990

(1991) Nogier JF. , Génération automatique de langages et graphes conceptuels, Paris : Editions Hermès.

(1991) HARNAD, Steven. Post-Gutenberg Galaxy: The Fourth Revolution in the Means of Production of Knowledge. Public-Access Computer Systems Review,, vol. 2, n° 1, p. 39 - 53.

(1992) Denhière G., Baudet S. , Lecture compréhension de texte et science cognitive, Presses Universitaires de France.

(1992) Fluhr C. , « Le traitement du langage naturel dans la recherche d'information documentaire » in Les interfaces intelligentes dans l'IST, Rapport INRIA, France.

(1992) Horacek H. , « An integrated view of text planning », Aspects of automated natural language generation, 6th International Workshop on Natural Language Generation, Trento, Italie.

(1993) Haton J.-P. et M.-C. , L'intelligence artificielle, 3ème édition corrigée, Collection Que sais-je ?, Presses Universitaires de France.

(1993) Balicco L. , Génération de répliques en français dans une interface Homme-Machine en langue naturelle, Thèse de doctorat, Université Grenoble 2, France.

Après 1995

- (1996) Ponton C. , Génération automatique de textes - Essai de définition d'un système noyau, Thèse de doctorat, Université Grenoble 3, France.
- (1996) Bisseret A., Montarnal C., « Linearization in spatial descriptions: Tour or hierarchical structures? » CPC, 15 (5), pp. 487-512.
- (1997) Pouchot S. , Etude de la structure de textes en langue naturelle en vue d'une application en génération automatique de textes, Mémoire de DEA, Université Grenoble 3, France.
- (1997) Balicco L. , « Génération automatique de descriptions textuelles de figures à deux dimensions », GAT'97, 1er colloque francophone sur la Génération Automatique de Textes, Grenoble, France.
- (1997) Ben Ali , Reformulation de réponses de systèmes documentaires bibliographiques, mémoire de DEA, Université Grenoble 3, France.
- (1997) BRUNET, Étienne. Les liens hypertextuels ou abondance de liens ne nuit pas, *Lexicometrica*,
- (1998) Leloup C. , Moteur de recherche et d'indexation : environnements client-serveur, internet et intranet, Paris : Editions Eyrolles.
- (1998) Mizzaro, S. , How many relevances in information retrieval?, *Interacting with computers*, 10:3 305-322.
- (1999) Balicco L., Pouchot S. , « Analyses d'un corpus en langue naturelle pour la génération automatique de textes descriptifs », Atelier Corpus et TAL, TALN'99, Cargèse, France.
- (1999) Ben Ali S., Timimi I. , « De la paraphrase à la recherche d'information », CISI'99, ISD de Tunis, Tunisie.
- (1999) RASTIER, François, PINCEMIN, Bénédicte. Des genres à l'intertexte. *Cahiers de praxématique*,
- (1999) Bourdoncle F. , « Panorama et perspectives des outils de recherche d'Information textuelle sur Internet », Actes du colloque IDT'99, Paris, France.
- (1999) Sfez L. , La communication, 5ème édition corrigée, Collection Que sais-je ?, Presses Universitaires de France.

Après 2000

- (2000) Revelli C. , Intelligence stratégique sur Internet, Paris : Editions Dunod.
- (2001) KRICHEL, Thomas, WARNER, Simeon M. Disintermediation of Academic Publishing through the Internet: An Intermediate Report from the Front Line. Paper presented at the ICC/IFIP 5th Conference on Electronic Publishing in Canterbury, UK, ..
- (2001) GUÉDON, Jean-Claude. A l'ombre d'Oldenburg : Bibliothécaires, chercheurs scientifiques, maisons d'édition et le contrôle des publications scientifiques. ARL Meeting, Toronto, Mai.
- (2001) RASTIER, François. Humanités et sciences humaines. Texte diffusé sur la liste LaLif, vol. 1, n° 1
- (2001) RASTIER, François. Eléments de théorie des genres
- (2002) ANIS, Jacques. Communication électronique scripturale et formes langagières : chats et SMS. In Actes de la Quatrième rencontre Réseaux Humains / Réseaux Technologiques « S'écrire avec les outils d'aujourd'hui » Université de Poitiers.
- (2002) BADDELEY, Suzanne. Pratiques de l'écriture au cours des siècles. In Actes de la Quatrième rencontre Réseaux Humains / Réseaux Technologiques « S'écrire avec les outils d'aujourd'hui » . Université de Poitiers.
- (2002) PICOUET, Philippe, SAGLIO, Jean-Marc. Définition de parcours sur un Web Communautaire Paris : ENST, Rapport projet GET.

- (2002) ORASAN, Constantin, KRISHNAMURTHY, Ramesh. A corpus-based investigation of junk emails. Proceedings of The Third International Conference on Language Resources and Evaluation (LREC)
- (2003) CHARLET, Jean, LAUBLET, Philippe, REYNAUD, Chantal. Web Sémantique : Rapport final. Action spécifique 32 CNRS / STIC .
- (2003) CHARTRON, Ghislaine. Éléments pour une approche comparée de la publication scientifique. In @rchiveSIC Communication au Forum Universitaire « La communication scientifique en quatre dimensions ». 20 mai 2003. Localisation par domaine : Edition électronique.
- (2003) MOIRAND, Sophie. Quelles catégories descriptives pour la mise au jour des genres du discours ? Journée d'étude « Les genres de l'oral », Université Lumière Lyon.
- (2003) REYNAUD, Chantal, CHARLET, Jean, LAUBLET, Philippe (coordonnateurs). Dossier Web Sémantique, Bulletin de l'AFIA
- (2003) KYHENG, Rossitza. La référence bibliographique : norme et praxis. In Texto Paris.
- (2003) RASTIER, François. De la signification au sens. Pour une sémiotique sans ontologie.
- (2003) PINCEMIN, Bénédicte. Rôle des ontologies pour le Web Sémantique : métadonnées ou données ?
- (2004) Journée d'étude ATALA « Le traitement automatique des nouvelles formes de communication écrite (e-mails, forums, chats, SMS, etc.) »
- (2004) RASTIER, François. Sciences de la culture et post-humanité. Conférence inaugurale du huitième congrès de l'Association Internationale de sémiotique, Lyon.
- (2004) RASTIER, François. Ontologie(s).
- (2004) BEAUVISAGE, Thomas. Sémantique des parcours des utilisateurs sur le Web. Texto ! Décembre
- (2004) SCHÖPFEL, Joachim, STOCK, Christiane. Grey Literature in an Open Context: From Certainty to New Challenges. In @rchiveSIC Communication au GL5 : Fifth International Conference on Grey Literature / 5ème colloque international sur la littérature grise.
- (2004) IACOVELLA, Andrea. Les portails de revues en Sciences Humaines et sociales. In @rchiveSIC . Lettre du département SHS du CNRS(69).

Récemment

- (2005) FORESTIER, Catherine. Les offres actuelles de livres électroniques pour l'enseignement supérieur : contenus, types d'accès, modèles économiques. In 2ème journée d'information-débat autour du thème : Le livre électronique dans l'enseignement supérieur : Quels usages ? Marseille, 2 mai
- (2005) GERINI, Christian. L'Open Access ou Paradigme de l'Accès Ouvert Electronique : les nouvelles technologies de l'information et de la communication au service d'une science libre et transparente. In @rchiveSIC 30 avril. Communication, Colloque International de Tunis, 14-16 avril 2005: L'information numérique et les enjeux de la société de l'information. Localisation par domaine : Édition électronique.
- (2005) LABBE, Hélène, MARCOCCIA, Michel. Tradition épistolaire et médias numériques : du billet au courrier électronique. In A. Betten & M. Dannerer (eds.), Dialogue Analysis IX - Dialogue in Literature and the Media / Selected Papers from the 9th IADA Conference, Salzburg
- (2005) PECCATTE, Patrick. Métadonnées: une initiation. In Web site de Soft Experience. Dernière mise à jour:
- (2005) SWAN, Alma, BROWN, Sheridan. Open access self-archiving: An author study. In Web site of JISC Committee for the Information Environment (JCIE), Scholarly Communications Group .
- (2005) VAILLANT, Pascal. : texte et toile - Quelques éclaircissements techniques et pratiques sur le Web et son utilisation

Annexes

Annexe 1

Vous cherchez un Prolog ? je vous conseille SWI Prolog (sous Linux ou windows), disponible gratuitement (licence GPL) au département informatique de l'Université de Psychologie d'Amsterdam (<http://www.swi-prolog.org/>)

Mais il en existe d'autres, en particulier GNU Prolog (par l'INRIA).

Prolog cherche à prouver que le but (goal) demandé est vrai. Pour cela, il analyse les règles, et considère comme faux tout ce qu'il n'a pas pu prouver. Quand le but contient une variable libre, Prolog la liera à toutes les possibilités :

```
?- est_un_homme(X).
X=marc
X=jean
?- est_le_pere_de(Pere,jean).
Pere=marc
?- est_le_mari(Pere,Mere), est_le_pere_de(Pere,jean).
Pere=marc, Mere=anne
```

On peut combiner les buts à l'aide des opérations logiques et (,), ou (;) et non (not). Le but défini ci-dessus nous permet de trouver la mère de jean. Mais on peut également créer une règle en rajoutant :

```
est_la_mere_de(Mere,Enfant) :-
    est_le_mari(Pere,Mere),
    est_le_pere_de(Pere,Enfant).
```

Désormais le but : *est_la_mere_de(Mere,jean)* (réponse : *Mere=anne*) permet de rechercher la mère d'un individu, mais la même règle permet de répondre également à une recherche d'enfants : *est_la_mere_de(anne,X)* (réponse : *X=jean*). On peut aussi vérifier une affirmation (*est_la_mere_de(anne,jean)*) ou afficher tous les couples de solutions à la requête : *est_la_mere_de(M,X)*. On ne trouvera ici qu'une solution, pour véritablement tester ceci il faut augmenter notre base de connaissances.

Avec les variables numériques

En Prolog, l'écriture : $X=Y$ peut avoir différents résultats :

- Si X et Y liées, répondra Yes si elles sont égales, No sinon.
- Si X libre et Y lié, proposera comme solution X valant la valeur de Y.
- Si X lié et Y libre, proposera comme solution Y valant la valeur de X.
- Si X et Y libres, erreur (car infinité de solutions)

Par contre l'écriture $X=Y+1$ (qui est équivalent à $Y+1=X$) va poser quelques problèmes. Il est tout d'abord obligatoire, dans cette écriture, qu'Y soit lié (Prolog ne sait pas inverser une équation, qu'elle soit simple comme ici ou plus compliquée).

Mais ce n'est pas tout : = va simplement recopier l'équation (en ayant instancié les variables) : $Y=23, X=Y+1, write(X)$. affichera $23+1$, et pas 24. C'est pourquoi deux autres opérateurs sont définis :

- `is` calcule (on dit aussi « évalue ») la valeur à droite (toujours liée) et instancie le résultat à la variable (libre) à gauche, ou la compare à la valeur liée de gauche. Par exemple `X is 10+15` met 25 dans X, ou dit Yes si X vaut 25 (parce que avant on a dit `X is 30-5` ou même `X=25`). Certains Prolog utilisent aussi l'opérateur `:=`
- `:=` évalue les expressions à droite et à gauche et regarde si les résultats sont égaux (les deux arguments doivent être liés)

exemple : écrire un programme qui pour un but « *affiche(10)*. » affiche les nombres de 10 à 0 : (on utilise le prédicat prédéfini `write`) :

```
affiche(X) :-
    write(X), write(' '),
    X>0, /* condition de fin */
    Y is X-1,
    affiche(Y).
```

Rq1 : il faut obligatoirement utiliser une variable intermédiaire (Y ici).

Rq2 : évidemment, là je triche quand même : c'est plus procédural que déclaratif.

Exo : afficher désormais dans l'ordre croissant .

Truc : il suffit d'inverser l'ordre des instructions, mais il répond false (comme d'ailleurs aussi dans le cas précédent) mais avant les affichages. Il suffit de donner le fait : `affiche(0)`.

Pour les autres comparaisons, les opérateurs (qui évaluent à droite et à gauche)

sont : `:=` `=\` (différent) `<` `>` `=<` `>=` , les opérateurs arithmétiques :

`+` `-` `*` `/` `mod` `**` , `sin`, `cos`, `log`,....

Annexe 2

Exemple de programme prolog simple :

```
est_un_homme(marc). /* l'argument est pour moi le sujet */
est_un_homme(jean).
est_le_mari_de(marc,anne)./* 1er arg: sujet, puis complément */
est_le_père_de(marc,jean).
```

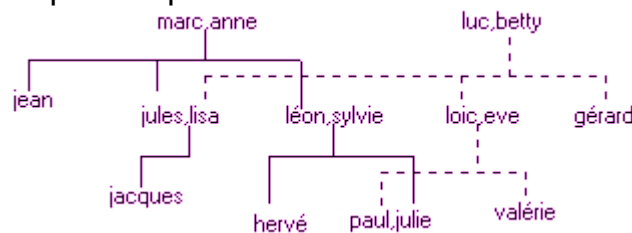
est_un_homme est un « prédicat ». Il sera soit vrai, soit faux, soit inconnu. Dans le « programme », toutes les règles concernant le même prédicat doivent être regroupées ensemble.

On peut alors « exécuter » le programme, en fait on fait une requête :

```
?- est_un_homme(marc).
Yes
?- est_un_homme(anne).
No
?- est_un_homme(luc).
No
```

Un exercice trouvé sur internet pour illustrer...

Compléter les données pour respecter :



On définira également les règles définissant : *est_le_fils*, *est_la_fille*, *est_la_belle_mere*,....

Pour effectuer un ou, on peut utiliser le OU (;) ou utiliser plusieurs règles:

```
est_la_belle_mere_de(BelleMere,Gendre) :-
    est_le_mari_de(Gendre,Epouse),
    est_la_mere_de(BelleMere,Epouse). /*Belle-mère d'un homme*/
est_la_belle_mere_de(BelleMere,Bru) :-
    est_le_mari_de(Epoux,Bru),
    est_la_mere_de(BelleMere,Epoux). /*Belle-mère d'une femme*/
```

On peut également définir les femmes par :

```
est_une_femme(F) :- not(est_un_homme(F)).
```

Cette règle marche pour des variables liées (dira Yes s'il ne l'a pas trouvé parmi les hommes) mais pas pour une variable libre (en effet, toute combinaison de caractères pourrait répondre à la requête). Pour qu'une règle puisse s'appliquer à une variable libre, il faut que l'on permette à Prolog de trouver toutes les possibilités (et donc qu'elles soient en nombre limité).

Exemple:

```
est_une_femme(F) :-
    est_le_pere_de(_,F), /* _ est le nom (imposé) d'une variable
                           dont la valeur ne nous intéresse pas, Prolog n'a
                           donc pas besoin de la lier */
    not (est_un_homme(F)).
```

Ici, on recherche d'abord ceux et celles qui ont un père, puis on élimine les hommes. On peut remarquer qu'anne et betty ne seront pas considérées comme femme. Il suffit pour résoudre ce problème de rajouter une seconde règle, traitant des épouses. Mais pour qu'il ne nous affiche pas deux fois celles qui sont à la fois épouses et filles, on pourra l'écrire ainsi :

```
est_une_femme(F) :-
    est_le_mari_de(_,F)
    and not (est_un_homme(F))
    and not (est_le_pere_de(_,F)) /* pour ne pas reprendre les cas
                                     traités par la règle précédente */
```

le but : *est_une_femme(X)* nous donne maintenant toutes les femmes que nous avons définies.

Exercice : rajouter tantes, cousins, grand mères...

Pour simplifier la gestion des hommes par exemple, on peut utiliser une liste :

```
hommes([marc, luc, jean, jules, léon, loic, gerard, hervé, jacques, paul]).
```

```
appartient_à(X,[X|_]).
```

```
appartient_à(X,[_|Queue]) :- appartient_à(X,Queue).
```

```
est_un_homme(X) :-
    hommes(Liste),
    appartient_à(X,Liste).
```

Annexe 3

Leurs évolutions syntaxiques et sémantiques

Les dates retenues correspondent au moment où le premier programme a pu être écrit et compilé dans le langage. Il est fourni quand c'est possible le site d'un compilateur ou interpréteur pour le langage ou une fiche détaillée pour les principaux d'entre eux. L'historique concerne surtout les langages d'application ou de script universels, mais je mentionne certains langages spécialisés importants et tous ceux qui ont eu une influence majeure dans la conception ultérieure d'autres langages de programmation.

Premier langage

Ada

Lovelace ainsi que Babbage et son neveu écrivaient des programmes pour le projet de machine à différences puis la machine analytique de Babbage.

En 1945, l'allemand K Zuse, inventeur de l'ordinateur Z3, aurait défini un langage évolué pour cette machine (avec arrays et records). On possède peu de documents sur ce langage.

Assembleur

Les assembleurs existent depuis le début des ordinateurs. Ils associent un nom symbolique au code du langage machine, par exemple:

```
add bx, 4      // Addition dans un registre
cmp [adr], 3   // Comparaison avec une variable
jmp address    // Saut à une adresse
```

La programmation en assembleur ne se pratique plus sur les ordinateurs actuels même pour les routines d'exécution rapides... Plusieurs langages actuels génèrent un bytecode portable qui est proche de l'assembleur, mais est invisible au programmeur.

Autocoder - 1952

Alick E Glennie

Implémenté d'abord sur Mark 1 puis sur d'autres machines, c'est un code symbolique qui se traduit en langage machine.

IPL - 1956

Information Processing Language

A Newell, H Simon, JC Shaw

Langage de traitement de listes, de bas niveau. Implémente la récursivité.

Fortran - 1954-1958

FORmula TRANslator system

Par John Backus et autres chercheurs d'IBM.

Langage dédié aux calculs mathématiques.

Fortran II en 1958 a introduit les sous-programmes les fonctions, les boucles, une

structure de contrôle FOR primitif.
Les identifiants avaient au plus six caractères.

Lisp - 1958-1960

LISt Processing

Mac Carthy

Langage fonctionnel de traitement de liste.

Il est récursif et non itératif. Les données et les programmes ne sont pas distingués et peuvent être traités de la même façon.

IAL - 1958

International Algebraic Logic

Premier nom d'Algol 58, non implémenté.

Algol - 1960 / Algol W - 1966 / Algol 68

ALGOarithmic Language

Défini par une commission internationale d'informaticien coordonnée par l'IFIP.

(pas de compilateur en distribution)

C'est le premier langage universel indépendant de la machine.

Introduit la grammaire BNF (Backus Naur Form) pour réaliser un parseur de syntaxe.

Introduit la structure de bloc d'instructions et les variables locales aux blocs.

Introduit la récursivité (malgré les réticences car on considérait cela comme superflu!).

Il utilise des tableaux dynamiques, les langages suivants comme Pascal et C ont donc régressé en utilisant des tableaux statiques pour des raisons de performance.

On y trouve le IF THEN ELSE, un FOR assez général, le symbole d'affectation :=, un SWITCH avec des gotos, les délimiteurs BEGIN END, le WHILE..

L'Algol W de Niklaus Wirth en 1966 introduit les RECORD, les déclarations de structures de données dynamiques, le CASE, le passage de paramètres par valeur, la précedence des opérateurs.

La même année, Niklaus Wirth crée Euler, langage intermédiaire entre Algol et Pascal.

Algol 60 restait un langage orienté vers le calcul mathématique. Pour revenir à l'objectif de langage général, une nouvelle version a été décidée en 1964, l'Algol X, devenu ensuite Algol 68.

Algol 68 utilisait += pour combiner l'affectation et l'addition. Il apportait la structure d'union et le cast de type. Il ajoute un IF THEN ELIF FI, le CASE, le GOTO, des opérateurs définissables par l'utilisateur.

Il ne permettait pas la compilation séparée de fichiers (dite incrémentale).

Cobol - 1960

COmmon Business Oriented Langage.

Défini par un comité, la CODASYL, COntference on DATA SYstems Languages.

Les travaux de la conférence, sous les auspices du Department Of Defense, avec des fabricants, uiversités et utilisateurs, ont duré de mai 1959 à avril 1960.

Grace Murray Hopper, qui avait défini Flow-Matic, un langage compilé dans les années 50, faisait partie du comité.

Langage procédural classique destiné à la gestion d'entreprise, dans lequel un programme est composé de 4 divisions: identification, environnement, data, procedure, qui peuvent comporter des sections. Il est fondé sur les données et défini précisément le matériel et les formats d'entrées et sorties de données.

Il introduit la structure de donnée RECORD. Les programmes sont auto-documentés par la syntaxe, ce qui ne les rend pas plus légers!

APL - 1964

A Programming Language

K Iverson.

Langage utilisant une notation mathématique, composé d'opérateurs. Un seul type, le tableau.

Défini entre 1957 et 1960 il a été implémenté en 1964.

Basic - 1964

Beginner's All-purpose Symbolic Instruction Code

John Kemeny, Thomas Kurtz

Il a été conçu, en 1963, pour être facile à apprendre, et implémenté en 1964. La première version était compilée, puis il est devenu interactif et interprété. Chaque ligne était numérotée pour permettre les branchements par GOTO!

Bill Gate et Paul Allen ont gagné un concours international en concevant l'implémentation d'un Basic performant, d'abord pour l'Altair (en 4 k. de mémoire) puis sur d'autre micro-ordinateurs.

Les micro-ordinateurs seront fournis avec un langage Basic en mémoire morte (ROM) jusqu'à la fin des années 80.

En 1977 les Apple II sont fournis avec un basic entier. Plus tard ils auront un basic Applesoft de Microsoft en virgule flottante. L'Applesoft à des identifieurs d'au plus deux caractères!, les branchements se font sur des numéros de lignes. Les sous-programmes sont appelés par un GOSUB sur un numéro de ligne.

Le premier PC d'IBM, sorti en 1981 utilise MS-DOS de Microsoft et son basic interprété (le Basica). En 82 Microsoft fournit un basic compilé (Quick Basic).

Pascal et le langage C se substitueront au Basic au cours de la même décennie, Microsoft fournit encore un basic compilé Visual Basic . ASP pour le Web et les langages d'extension d'application (macro) sont en basic.

True Basic, par les auteurs originels est compilé et n'utilise plus les numéros de lignes.

Iswim - 1965

If You See What I Mean. (Si vous voyez ce que je veux dire).

P. Landin

Premier langage fonctionnel, au sens mathématique du terme. Le premier aussi à utiliser l'évaluation paresseuse (lazy evaluation).

Grammaires Attribuées - 1965

Donald Knuth

Complétant la méthode BNF, les grammaires attribuées (attribute grammar) décrivent la sémantique des langages sous forme de fonctions exécutables. Ce type de grammaire facilitera la réalisation de compilateurs.

Simula 67 - 1962-67

Ole-Johan Dahl, Kristan Nygaard

Le projet Simula a démarré en 1962. Le but était d'en faire un outil de description de système d'évènements discrets ou de réseau, et un langage de programmation de simulation.

Il était conçu comme une extension au langage Algol.

En 1964, Simula 1 était implémenté sur Univac 1107. Il a été utilisé alors pour contrôler des administrations, des aéroports, de la planification, du transport, ou un système social.

C'était alors un outil spécialisé. En 1966 a été prise la décision d'en faire un langage universel. Plusieurs projets ont été lancés avec des constructeurs d'ordinateurs différents (Ibm, Univac, Digital) qui ont abouti à Simula 67.

Ce langage universel introduisait la notion de classes, de sous-classes et d'objets instances des classes. Les classes permettent d'associer les fonctions (méthodes) aux objets.

Logo - 66

W Fuerzeig, S Papert, et autres.

Destiné à apprendre la programmation aux enfants, il est proche de Lisp, et basé sur le déplacement d'une « tortue » sur l'écran.

Snobol 4 - 1967

StroNg Oriented symBOLic Language

D.J Farber, R.E. Griswold, F.P. Polensky au Bells Labs

Snobol est apparu en 1962.

Snobol 4 est la première version stable et distribuée de Snobol, en 1967.

C'est un langage de traitement de texte ou manipulation de chaînes de caractères, basé sur le principe de patterns, concaténation et alternation.

Il utilise essentiellement des tableaux et des tables. C'est le premier langage à implémenter des tableaux associatifs (dictionnaires) indexés par des clés de tous types.

Il est aussi le premier à implémenter le pattern-matching, sorte de switch case

élaboré.

On peut , exécuter du code contenu dans des chaînes de caractères..

Les types sont: string, integer, real, array, table, pattern et types définis par l'utilisateur.

CPL

Combined Programming Language.

Universités de Cambridge et Londres.

C'est un mélange d'Algol 60 et de langage fonctionnel destiné à calculer la preuve de théorèmes. Il utilisait des structures de test polymorphiques. Langage typé avec un type joker « any ». Types structurés liste et table.

Complexe, il n'a pas été implémenté. Je le cite seulement parcequ'il a été un pas vers le langage C.

BCPL - 1965

Basic CPL.

Martin Richards

Ce langage se voulait une version simplifiée du CPL.

Il utilise les constructs FOR, LOOP, IF THEN, WHILE, UNTIL, REPEAT, REPEAT WHILE, SWITCH CASE, etc...

Il possède des procédures et des fonctions contrairement à C.

Les délimiteurs de block sont les symboles \$(..... \$) qui ont sans doute inspiré les délimiteurs de commentaire du C: /* ... */

Pascal - 1970

Du nom de Blaise Pascal, mathématicien français.

Niklaus Wirth.

Langage conçu pour faciliter la réalisation des compilateurs et qui dirige l'enseignement de la programmation en obligeant à une programmation structurée.

L'UCSD Pascal est la première version sur micro ordinateur, réalisé par un groupe de programmeurs dirigé par Kenneth Bowles. Il compile les programmes en P-code, qui est interprété et portable (comme plus tard Java). Il comporte un environnement de développement complet, idée reprise avec succès par Turbo Pascal.

En 1981, un jeu de rôle écrit en Pascal, Wizardry, connaît un grand succès sur Apple II.

C'est avec l'apparition de Turbo Pascal en 1983 (Anders hejlsberg), rapide et doté d'une IDE complète que le langage s'est répandu.

Les constructs sont proches du C. La façon de déclarer les variables plus lourde sans être forcément plus logique.

Icon - 1970

Griswold

Langage procédural, avec fonctions de traitement de texte comme Snobol4, et des constructs puissants.

Il dispose de types structurés: liste, set, table (dictionnaire).

Une liste s'écrit: nom := [mot,mot, nombre, etc...]

Une liste s'adresse avec un indice, comme a[i] ou par une gestion de pile.

Un set contient des éléments non doublés et dispose de fonctions d'union, intersection, suppression.

Le construct range s'écrit: a to b

ou "to" est le mot-clé et "a" et "b" sont des variables..

On peut placer un range dans une expression ou comme argument d'une fonction.

L'expression et la fonction seront appelés pour chaque valeur de l'intervalle.

Ex: write(1 to 5) affichera 1, 2, 3, 4 ,5.

Les expressions ne retournent pas une valeur vraie ou fausse mais sont évaluées ou rejetée. Les opérations associées sont exécutées quand l'expression est évaluée. C'est le premier langage à évaluation des expressions dirigée par le but.

L'instruction du langage C: si (x= expression) ... qui signifie: assigner à x le résultat de l'expression, et si a est différent de zéro, alors..., cette construction est généralisée en Icon, si l'expression est évaluée, x prend sa valeur et la condition est remplie, sinon x n'est pas changé et la condition est passée.

Le construct every ... do associé à une expression fonctionne comme un itérateur. De même que le range.

L'alternation est un autre construct original puissant. Il permet d'utiliser une succession de paramètres, éventuellement jusqu'à obtention d'un résultat. Son symbole est "|".

Par exemple l'instruction: si a | b | c = 0 | 1 alors

se lit: si a ou b ou c vaud 0 ou 1 alors ...

Forth 1971

Fourth réduit à Forth pour la contrainte de 5 lettres de l'IBM 1130.

Charles H Moore

Définit dans les années 60, implémenté apparemment en 71.

Langage d'Astronomie qui utilise une pile à la place des variables.

Il se voulait langage de 4 ième génération, d'ou le nom.

Smalltalk - 1972

Alan Kay et le Software Concept Group.

C'est un langage totalement orienté objet qui fonctionne à l'intérieur d'un environnement graphique, avec fenêtre, la souris. La notion de bitmap (1 pixel = 1 cellule de mémoire) est introduite.

C - 1973

C succède à B, qui succède à BCPL.

Par Dennis Ritchie.

C'est un langage destiné au départ à programmer le système d'exploitation UNIX, et qui est devenu rapidement universel grâce à sa portabilité et ses performances..

Il permet la compilation de fichiers séparément.

En 1965, les programmeurs d'ATT utilisent le langage Bcpl pour travailler sur la réalisation d'Unix. Insatisfaits du langage, ils l'ont fait évoluer dans une nouvelle version appelée B, puis dans un nouveau langage appelé C qui obtiendra le succès que l'on sait.

C'est surtout l'évolution du matériel qui a incité à créer le C. Les langages Bcpl et B utilisaient des entiers comme pointeurs, mais sur des machines conçues différemment, cela ne se pouvait plus.

Bcpl n'avait pas de type. Les déclarations du genre `int i`, `char b` ont été créés avec C. D'autres types sont apparus ensuite.

L'opérateur `+=` vient d'Algol 68, mais cela s'écrivait plutôt `=+`

Bcpl plaçait un bloc d'instructions entre `(*` et `*)` comme un commentaire l'est entre `/*` et `*/` et une sous-expression entre `(` et `)`. Je suppose que ce symbolisme veut dénoter le fait que que toute chose est une expression dans le langage, tout en accélérant le parsing. Le langage C simplifie avec les symboles `{` et `}`, ce qui enlève le sens originel. Les notions d'union et cast viennent d'Algol 68.

L'opérateur `++` existait dans le langage B.

La directive "include" vient du PL/1.

Le préprocesseur à été implémenté en 1973 et l'utilisation effective à commencé, donc c'est à cette date que je place la création du langage C, même si la maturation à commencé à partir de 1969. Le langage à continué d'évoluer jusqu'en 1980. A partir de 73, C à été utilisé pour programmer le noyau d'Unix.

Prolog - 1970+

A. Colmerauer, D. Roussel.

Inria

Le langage à été développé conjointement en France à Aix-en-Provence et à Edimburg.

Il introduit la programmation logique. Un programme est composé de clauses de Horn.

Prolog se dit déclaratif parce que son système d'inférences logiques constitue un mécanisme de résolution.

Sql - 1970+

Standard Query Language

IBM

mySQL

Langage d'interrogation de bases de données relationnelles. Il succède au langage Square.

Awk - 1974

Selon les initiales des auteurs.

Aho, Kerningham, Weinberger

Langage de traitement de texte basé sur des expressions régulières, fonctionnant selon le principe pattern-action.

Scheme - 1975

De "schemmer"

MIT - G Steele, G Sussman

Langage dérivé de Lisp, utilisé comme langage de script.

Plasma - 75

Carl Hewitt

Langage à acteurs. Implémenté en Lisp.

Sasl - 1976

Saint Adreus Static Language

D Turner

Destiné à apprendre la programmation fonctionnelle.

Descend de Iswim, structures de données en nombre illimité.

ML - 1973?

Meta Language

R Milner

Langage fonctionnel inspiré de Iswim.

Il avait pour but la preuve de théorème à l'université d'Edimbourg.

Les fonctions sont remplacées par des pattern models.

Implémenté en Lisp.

Modula 2 - 1979

MODUlar LAnguage.

Niklaus Wirth

Modula 1 aurait été défini en 1977. Implémenté sur station de travail Lilith à l'origine.

L'idée du langage est de réduire le risque d'erreur avec des règles de programmation coercitives. Cependant, il se rapproche du langage C en tentant de combler les lacunes de Pascal. Ainsi, un appel de fonction sans paramètre s'écrit f() comme en C et non f comme en Pascal.

Il découpe un programme en modules contenant des routines et des structures de données, avec une visibilité locale, et avec des interfaces entre modules. Utilise des coroutines. Il apporte des fonctions d'accès au hardware pour concurrencer le C.

Il sera peu utilisé hors du cadre universitaire, parce que ces améliorations (modules, accès matériel), ont été ajoutés aux distributions de Pascal (avec notamment les units de Turbo Pascal).

Ada - 1980+

Du prénom de Ada Byron de Lovelace, première femme à programmer.

Créé par un groupe de travail dirigé par Jean Ichbiah, sur un cahier des charges du Département de la défense des USA.

Inspiré par Pascal et Algol W.

Introduit la généricité des algorithmes et une sorte d'orientation objet primitive, il deviendra orienté objet par la suite après C++.

Introduit les paquetages, modules indépendants.

C++ 1981-1986

Bjarne Stroustrup.

Langage orienté objets, selon le principe de Simula.

Introduit la surcharge des opérateurs. Les méthodes peuvent être inline.

A coté du symbole /* et */ pour enclore les commentaires, il utilise le symbole // pour un commentaire d'une ligne. On notera la réapparition d'un symbolisme qui existait déjà dans le langage Bcpl, auquel le langage C a succédé!

Objective C, inventé par Brad Cox en 1984 est une autre version orientée objet de C qui s'inspire de smalltalk. Pas de surcharge des opérateurs.

Utilisé sur le défunt ordinateur Next et pour réaliser le système d'exploitation NextStep.

Standard ML - 1984

R Milner, université d'Edimburgh et Cambridge, Inria.

Rechercher "Standard ML Moscow" sur moteur de recherche.

Implémentation de ML.

Eiffel - 1985

Bertrand Meyer

Langage procédural totalement orienté objet, implémente la persistance conçu pour la sécurité du logiciel.

Se compile en C. Peut s'interfacer avec d'autres langages. Il incorpore des éléments de langage fonctionnel, dispose de classes génériques, d'un garbage collector.

Une version dérivée existe en open source, Sather, (nom de la tour de Berkeley).

GAP - 1986

Groups, Algorithms and Programming

Groupe d'étudiants allemands, incluant Johannes Meier, Werner Nickel, Alice Niemeyer, et Martin Schönert

Gap Software

Le langage a été défini pour programmer des algorithmes mathématiques.

Il est interprété, interactif et non typé. Liste et records les structures de base.

La syntaxe s'inspire de Pascal avec des différences. Les commentaires sont insérés avec #.

Une fin de bloc est notée par inversion des mot-clés: if fi, do od.

La boucle for à la forme for in liste ou for from to.

Le langage distingue procédures et fonctions.

Ce qui le caractérise est que les variables pointent sur une valeur et non une adresse de mémoire, et la définition d'une fonction qui a la forme d'un appel: `x := fonction (arguments) bloc`.

On peut imbriquer une fonction dans une autre fonction

Miranda - 1989

Du nom d'une héroïne de Shakespeare (Miranda, admirable en latin).

D.Turner

Inspiré de Sasl et ML. Les arguments d'une fonction ne sont évalués qu'au moment où ils sont utilisés (lazy evaluation). Pattern-matching imbriqué, modules.

Caml 1987

Categorical Abstract Machine Language

Suarez, Weiss, Maury

Inria

Caml et Objective caml en 1996, implémentent ML.

Perl - 1987

Practical Extracting and Report Language.

Par Larry Wall, linguiste australien.

Perl

Destiné à remplacer les langages de ligne de commande d'Unix, Sh et Sed ainsi que Awk, il reprend la même (affreuse) syntaxe. Sert surtout à l'administration de système, et aux scripts CGI.

Utilise listes et tableaux associatifs (dictionnaires). Le construct FOREACH permet de parcourir le contenu de listes.

Oberon - 1988

Niklaus Wirth

Après Modula 2, l'auteur définit le langage Lilith en 1980 (sur machine Lilith), puis Oberon.

Plusieurs constructs d'usage commun sont supprimés afin de réduire encore le risque d'erreurs! Un garbage collector est ajouté.

Haskell - 1990

Prénom du logicien Haskell Curry.

Langage élaboré par un comité pour les langages fonctionnels.

Haskell

Langage purement fonctionnel. Inspiré de Miranda, donc de Sasl.

Tableaux fonctionnels, pattern matching.

ABC 1980-90

ABC selon l'expression.

Langage de script mis au point par CWI aux Pays-bas et qui se voulait un successeur du Quick Basic ou des langages de script sous Unix.

Sans doute le premier à utiliser l'indentation seule pour dénoter l'imbrication des procédures: il n'y a pas de délimiteurs comme begin/end ou équivalent.

Autre innovation, il n'y a pas de gestion de fichier, mais persistance des variables globales: elles conservent leur contenu d'une session à l'autre!

Il y a cinq types de base: nombre, chaîne, liste, composé (structure sans noms de champs), tableau.

Python - 1991

Le nom vient de la série de télévision anglaise "Monty Python Flying Circus"

Guido Van Rossum

Langage de script au typage dynamique. C'est une alternative à Perl.

S'inspire de ABC mais est extensible avec des modules en langage C et orienté objet.

Comme ABC il utilise des types évolués: tuple, liste, dictionnaire.

L'opérateur slice [a : b] permet d'extraire un sous-ensemble d'une liste.

Il existe une version qui compile en bytecode Java, jython et une version pour .NET sur Active State.

Pov-Ray - 1991

Persistence Of Vision (titre d'un livre médiocre de science-fiction).

PovRay

Pov-Ray est un langage de description d'images.

DisCo - 1992

Distributed Co-operation

Reino Kurki-Suonio

Disco

Disco est un langage de spécification pour les systèmes réactifs, avec la syntaxe de Pascal. Les constructs du langage sont les objets, les fonctions commandées par événements (appelées ici actions) et les relations. Une fonction est activée quand un état du système survient et peut-être surchargée. Disco est focalisé sur les interactions collectives. Les layers (couches, cadres) sont les modules du langage. C'est un langage orienté-système avec objets et réactions (et non orienté-action comme il est dit dans la présentation).

Ruby - 1994

Comme la pierre précieuse, par analogie avec Perl

Yukihiro Matsumoto

Ruby-Lang

Populaire au Japon, Ruby a été conçu comme successeur à Perl et Python pour être plus clair que le premier et plus orienté-objet que le second. La syntaxe s'inspire de ces deux langages, elle se veut intuitive et naturelle, mais peut être complexe.

Il n'y a pas de nouvelles structures de contrôle comme en Scriptol, mais une quantité

d'innovations mineures qui réduisent la taille du code.

C'est un langage interprété, facilement extensible. Les instructions sont terminées par les fin de lignes. Les blocs d'instructions et les boucles sont terminées par "end". On retrouve la plupart des caractéristiques de Python: tableaux associatifs, iterateurs...

Son originalité est dans les objets dynamiques (ajout de méthodes aux instances) et dans le scope des variable dénoté par un préfixe.

Java - 1994

Java se traduit par caoua, café.

James Gosling et autres programmeurs chez Sun.

Conçu à l'origine, en 1991, comme un langage interactif, et nommé Oak, il n' aucun succès. Mais en 1994 il est réécrit pour Internet et renommé Java (café, caoua). En 1995 les navigateurs peuvent exécuter des applettes. En janvier 1996, Javasoft sort JDK 1.0, le Java Développement Kit.

Java est un langage procédural classique inspiré du C++. Il se compile en bytecode, interprétable sur tout ordinateur. (Visual Café et GCJ produisent du code machine). Il simplifie le C++: une classe par fichier, gestion automatique de la mémoire, pas de pointeurs. Il le restreint: pas d'héritage multiple ni de surcharge des opérateurs, mais il ajoute le multitâche intégré, la portabilité.

Java n'a que des tableaux dynamiques, au contraire de C et C++.

PHP - 1995

Personal Home Pages Hypertext Processor

Rasmus Lerdorf

Langage de script multi-plateformes, s'intégrant au Html.

Similaire au langage C, non typé, les variables sont préfixées par le symbole \$ comme sous le shell Unix ou Perl. Il parse des pages html incluant du code php et délivre une page en pur html.

Une librairie de fonctions étendue permet au Webmaster du créer des pages dynamiques et interactives.

Microsoft utilise sous Windows un langage équivalent, l'ASP, proche du basic.

UML - 1996

Unified Modeling Language

Standard par OMG (Object Management Group) - Grady Booch, Jim Rumbaugh, and Ivar Jacobson

Ressources

Uml est l'union de trois langages de modélisation conçus par les trois auteurs ci-dessus. Le langage utilise une notation graphique pour décrire des projets logiciels. Un source est un diagramme représentant des objets et leur interactions. Un modèle est fait de vues et leur combinaison décrit un système complet. Le modèle est abstrait et indépendant du domaine.

ECMAScript - 1997

ECMA Script

Netscape - Standard from ECMA

More

Langage de construction de pages html dynamiques, coté client.

Inventé par Netscape.

Il est similaire à Java mais non typé et plus puissant, avec des caractéristiques communes avec Php.

Rebol - 1997 (La definition est plus ancienne)

Relative Expression-Based Object Language

Carl SassenRath

Rebol

Langage de scripts interprété utilisant un code compact. Il est destiné la communication sur Internet et les systèmes distribués. Il est extensible.

Comporte 45 types utilisant les mêmes opérateurs (ext: date, monnaie...). Les blocs d'instructions sont enclos entre [].

C# - 2000

(C-sharp), note de musique, succède à C++ (synonyme de mélodie?)

Anders Hejlsberg / Microsoft.

Ce langage est promis à un grand succès à partir de 2002. Il est le langage de base de la plateforme .NET, pour programmer des logiciels utilisable à distance sur Internet notamment. Comme Java, il reprend la syntaxe du langage C (qui a quand même 30 ans!) avec les mêmes simplifications: garbage collector, absence de pointeur, pas d'héritage multiple, interfaces, multi-tâche...

Le C# se compile en langage intermédiaire, le MSIL (MicroSoft Intermediate Language), et fonctionne avec une bibliothèque multi-langages, le CLR (Common Language Runtime). L'originalité essentielle est que des langages différents peuvent être compilés en MSIL et partager leurs classes.

D'autres innovations ont été incluses au langage:

- les structs sont ici des objets spéciaux passés par valeur.
- Les littéraux sont des objets, avec des méthodes..
- Les attributs sont des objets descriptifs attachés aux éléments du programme et utilisés par le runtime.
- Les propriétés: une méthode définie comme propriété est utilisée comme une variable: prop = 5 est équivalent à prop(5).
- Le construct foreach() pour parcourir des tableaux (nouveau seulement par rapport au Java et au C++).
- Le delegate qui remplace le pointeur de fonctions du langage C.

Par rapport au Java, outre les différences ci-dessus:

- La gestion d'évènements est mieux intégrée.
- Il conserve la surcharge des opérateurs du C++
- Accès plus simple au système natif..

AspectJ - 2001

Aspect for Java

Palo Alto Research Center

Aspect J est une extension Java qui implémente la programmation orienté-aspect. Une technique qui modularise les "concerns" transversaux. Ici l'unité n'est pas la classe, mais un concern, qui se partage entre plusieurs classes. Les concern peuvent être des propriétés, des zones d'intérêt, d'un système et la POA décrit leurs relations, les compose ensemble dans une programme. Les aspects encapsulent un comportement commun à plusieurs classes.

Scriptol - 2001

Scriptwriter Oriented Language

Denis. Sureau

Le plus récent, le plus puissant des langages de programmation procéduraux. Scriptol peut-être compilé en PHP, en C++ ou en exécutable, ce qui lui donne une grande portabilité. C'est à la fois un langage d'applications, de scripts et de pages Web. Les blocs d'instructions et les structures de contrôle ne sont pas clos par "end" ou "}" mais, comme xml, avec la forme: "/if", "/for", " /while", etc...

Le langage dispose de nouvelles structures de contrôle: "for in", "while let", "scan by", etc... Le "if composite" facilite l'implémentation de règles.

Les variables et les littéraux sont des objets. On crée un objet primaire (nombre, texte, etc...) ou évolué par affectation directe d'une valeur ou d'une liste d'arguments au nom.

Scriptol est destiné à évoluer pour fournir, à côté des classes, d'autres structures de haut niveau et rapprocher la programmation de la pensée.

Depuis octobre 2003, scriptol permet d'utiliser Xml comme structure de données interne.

E4X - June 2004

EcmaScript For Xml (E four X)

ECMA

Ecma

E4X n'est pas un langage, mais une addition à un langage. De la même façon que Scriptol utilise Xml comme structure de données avec les instance de son propres Dom, E4X permet d'assigner Xml aux variable EcmaScript, et d'accéder aux élément par un indice ou un attribut.

Annexe 4

Un extrait de présentation faite par les membres du laboratoire ICM de l'Université Stendhal (Canada) du générateur de texte CRISTAL, destiné à la recherche d'information

« [...]Voici la présentation d'un **générateur du français écrit**, implémenté en Prolog. Cet outil est non contextuel (Balicco 1993, Ponton, 1996) c'est-à-dire indépendant de toute application.

Il fonctionne sur la partie « comment le dire ? » et est donc fondamentalement un générateur de surface traduisant sous forme linguistique un contenu sémantique donné.

Notre système est basé sur un modèle linguistique qui permet la **production de plusieurs versions « équivalentes » d'un même contenu sémantique**. Ce modèle organise le texte à l'aide d'opérations linguistiques qui servent à l'élaboration de toutes les versions différentes du texte.

Nous appelons opération linguistique la mise en œuvre d'un phénomène linguistique. Deux types de phénomènes linguistiques sont considérés par notre générateur : les phénomènes de regroupement (coordinations, relatives...) (Dale 1989), (Horacek 1992) et les phénomènes référentiels (anaphores, ellipses...) (Danlos 1989), (Nogier 1991).

Nous utilisons des opérations de regroupement pour produire des modifications de structure en regroupant des éléments qui peuvent être associés, entre des propositions différentes, par exemple. Une opération linguistique référentielle remplace des éléments textuels par des anaphores telles que les pronoms ou les ellipses. Par exemple, on peut remplacer des syntagmes nominaux par des pronoms personnels, ce qui est un problème étudié dans différents systèmes de génération comme ceux décrits dans (Danlos 1989) ou (Nogier 1991).

Qu'elles soient de regroupement ou référentielles, ces opérations doivent d'abord être détectées à partir du contenu fourni en entrée (sous forme prédicative). Le générateur passe donc en revue les différentes propositions et recherche si des

opérations de regroupement (coordination par exemple) ou référentielles (telles que l'anaphorisation) sont possibles entre deux propositions.

Pour cela, il a recours à des règles de détection formant le modèle linguistique et constitue ainsi un ensemble d'opérations linguistiques applicables sur le texte. Le système choisit ensuite de réaliser certaines d'entre elles. Ce choix est paramétrable et permet donc de modifier la composition linguistique des textes.

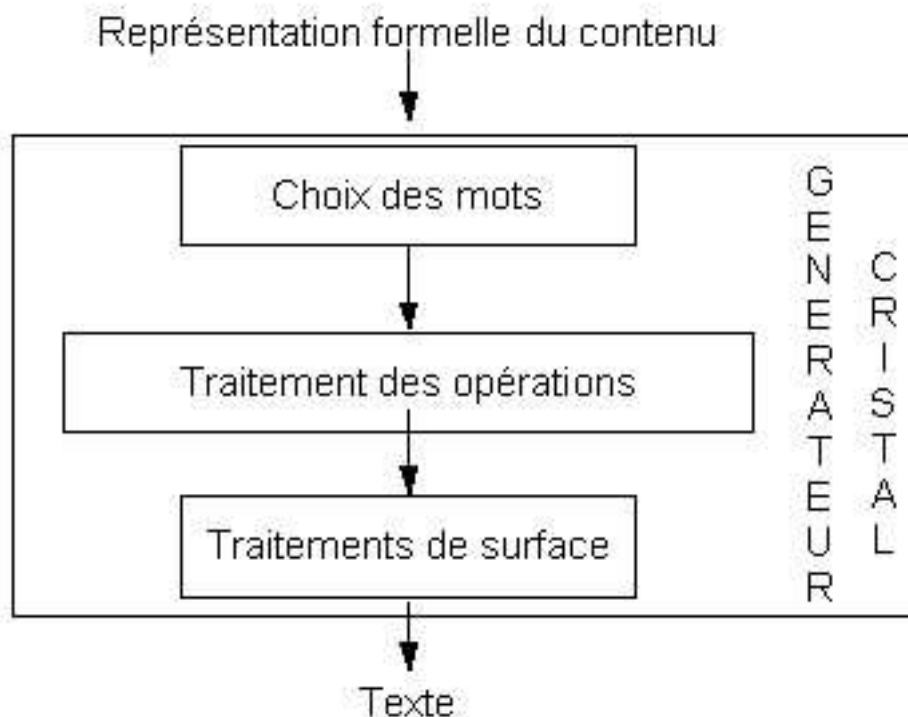


Fig. 1. Fonctionnement du générateur CRISTAL

Comme le montre le schéma ci-dessus, le processus de génération est décomposé en plusieurs tâches qui sont définies dans des modules distincts. Ils peuvent donc être modifiés indépendamment sans conséquence sur le fonctionnement général du générateur. Le système commence par le choix des mots (un message sous forme logique est transformé en unités lexicales), puis la détermination de la structure syntaxique à l'aide des opérations linguistiques. La génération se poursuit par une phase de génération morphologique (calcul des formes fléchies, conjuguées...) avant

de terminer par des traitements de surface (majuscules, élisions...). Les opérations linguistiques sont concrètement réalisées lors de l'écriture du texte. Pour cela, des règles de transformation sont appliquées à la représentation formelle du contenu du texte. **Le générateur utilise des ressources linguistiques indépendantes de l'application (un lexique et une grammaire) et un lexique propre à chaque application.**

Il nous a semblé important de tester et valider cet outil. Pour cela, nous lui avons associé une première application consistant à produire des textes équivalents à ceux d'un corpus en langue naturelle (Balicco 1997). Ce corpus a été recueilli par une équipe de l'INRIA. Plusieurs raisons ont guidé ce choix.

Tout d'abord, ce corpus a été rassemblé dans le but d'étudier les processus de linéarisation et les stratégies lors de la production de textes descriptifs. Cette tâche repose sur des processus cognitifs complexes. Une étude des textes conçus dans ce cadre fournit donc des bases solides pour notre objectif de simulation automatique de ces processus (Balicco, Pouchot 1999), tout travail d'écriture nécessitant une phase préalable de linéarisation. Ce qui est observé chez les sujets humains permet de mieux appréhender la génération automatique.

Ensuite, le monde auquel il est fait référence est assez restreint, donc plus facilement maîtrisable ; les énoncés descriptifs sont relativement courts mais riches et extrêmement variés en terme de vocabulaire, de structures logiques, de mises en forme matérielles (Pouchot 1997).

Enfin, reproduire un corpus permet également de comparer les résultats obtenus (textes engendrés) aux textes naturels. Le processus est donc bouclé puisque le principe est de partir de textes écrits par des sujets, d'en extraire des modèles pour les intégrer au générateur afin qu'il produise des textes, destinés à être lus, compris et éventuellement analysés par des humains (Denhière, Baudet 1992).

La génération automatique de textes

La génération de textes consiste à faire produire des textes par un ordinateur, de façon à exprimer automatiquement un contenu formel en langue naturelle. On

partage traditionnellement la génération en deux étapes : le « quoi dire ? » qui consiste à déterminer le contenu du texte à engendrer, et le « comment le dire ? » qui est son expression en langue naturelle. De plus, l'objectif est l'élaboration de textes linguistiquement corrects et compréhensibles par le destinataire.

La qualité des textes engendrés, mais surtout leur compréhension sont fonction des informations dont le générateur dispose pour produire ces textes. Il devient alors essentiel de recenser toutes les connaissances nécessaires au générateur, afin de les lui fournir ou de lui donner les moyens de les calculer.

En effet, les textes élaborés dépendent d'abord des informations que l'on souhaite transmettre au destinataire du message. Ces éléments varient en fonction du contexte et donc de l'application visée. Ils sont représentés formellement (concepts, graphes ou autres) et constituent le «*quoi dire ?* ». D'autres informations plus contextuelles sont indispensables pour passer du «*quoi dire ?*» au «*comment le dire ?*». Elles sont en partie données et en partie calculées au cours du traitement pour guider le générateur tout au long de sa tâche.

Les informations manipulées par le processus de génération sont donc de types très divers :

- générales : on parle aussi de connaissances universelles,
- sur l'application : explicites ou implicites (on choisit de détailler plus ou moins une information),
- linguistiques : essentiellement morphologiques et syntaxiques, mais aussi sémantiques,
- sur le destinataire du message, avec le principe d'un « modèle » décrivant le destinataire.

La génération : une aide à la recherche d'information

L'application précédemment décrite nous a permis de valider le générateur et d'y intégrer de nouvelles données pour en améliorer la production. Cette application a également mis en avant la nécessité de prendre en compte l'utilisateur. Il s'agit ici de faciliter la recherche de l'utilisateur par l'association de notre générateur à un

système de Recherche d'Informations. La première partie de notre étude porte sur la problématique de l'interaction système de Recherche d'Informations/utilisateur. La seconde s'intéresse aux apports de la génération de textes à la Recherche d'Informations.

Problématique de l'interaction homme-système en recherche d'information

Trois acteurs principaux interagissent lors d'une phase de recherche d'information : l'utilisateur, le Moteur de Recherche (MR) et la Base de Données (BD). Le MR et la BD forment le Système de Recherche d'Informations auquel l'utilisateur accède via une interface. L'interaction entre ces trois acteurs se situe à différents niveaux. Pour répondre à un besoin réel d'information (RIN) auquel correspond un ensemble (éventuellement vide) d'informations pertinentes (IP) dans la base de données, l'utilisateur interroge le SRI. Pour cela, il construit une représentation mentale (PIN) de son RIN et l'exprime sous forme de requête à destination du SRI. Après le traitement de la requête par le MR en relation avec la BD, un ensemble de réponses (éventuellement vide), défini ici comme Résultat de la Requête (RR), est envoyé à l'utilisateur. La pertinence et l'exhaustivité de ces réponses dépendent de plusieurs facteurs ; elles sont de plus soumises à l'appréciation de l'utilisateur.

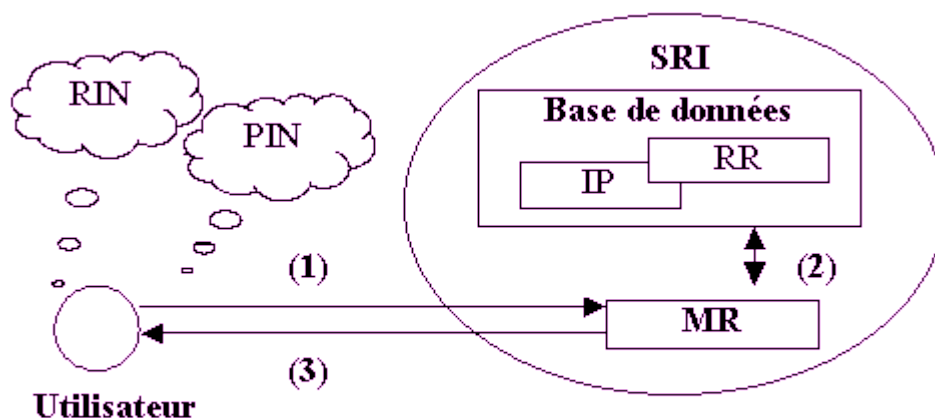


Fig. 2. Interactions en recherche d'information

L'un des problèmes de la recherche d'information consiste à obtenir le meilleur recouvrement entre l'ensemble des informations pertinentes et les résultats. Ce problème a deux origines.

- *Le passage du RIN au PIN.* La complexité de cette opération est essentiellement due au fait que l'utilisateur recherche des informations sur lesquelles il possède un état de connaissance justement incomplet. Il ne peut, ainsi, avoir qu'une perception incorrecte de son RIN. Ceci se reflétera sur son PIN et par transitivité sur sa requête.
- *Le passage du PIN à la requête.* Lors de la formulation de la requête, l'objectif est d'atténuer dans la réponse la production de « bruit » (réponses non pertinentes) et la production de « silence » (manque de réponses pertinentes). Afin de réduire la portée de ce double problème, la requête de l'utilisateur doit être adaptée aussi bien à la base de données qu'au moteur de recherche. A cette fin, seul l'utilisateur peut modifier sa requête initiale. Cependant, certains systèmes proposent des aides à l'utilisateur pour mener à bien cette modification. Par exemple, des systèmes comme AskJeeves proposent à l'utilisateur différentes reformulations de la requête initiale. D'autres comme DigOut4U utilisent un lexique de synonymes pour aider à reformuler la requête.

Les réponses du système constituent un deuxième écueil pour la recherche d'information. En effet, elles doivent être en adéquation avec la requête et convenir à l'utilisateur. Si l'adéquation réponses/requête est un phénomène dû au moteur lui-même, il n'en est pas de même pour l'adéquation réponses/utilisateur. Par exemple, un ensemble important de réponses pertinentes par rapport à la requête mais mal présenté risque de nuire à la compréhension de l'utilisateur ou, du moins, de lui compliquer la tâche. Un autre exemple est la production de réponses sous forme de listes alors que l'utilisateur attend un texte simple. Plusieurs systèmes prennent en compte la présentation des résultats et y apportent diverses solutions. Certains, comme Copernic ou BullEyes, permettent différents tris des listes de résultats.

D'autres, comme DigOut4U ou Inforian Quest, peuvent, à la demande, produire un court résumé des réponses.

Un troisième problème à résoudre est lié à l'appréciation des réponses par l'utilisateur. En effet, s'il est seul à pouvoir déterminer les réponses pertinentes parmi celles proposées, il ne possède, par contre, aucun critère lui permettant d'évaluer leur nombre. Dans le meilleur des cas, le système propose des réponses mais comment mesurer leur exhaustivité ? Dans le cas inverse, le système ne fournit aucune réponse :

- les informations pertinentes existent dans la base de données mais la requête de l'utilisateur n'est pas adaptée au MRI,
- aucune information pertinente n'existe dans la base de données. Ici l'utilisateur ne sait généralement pas à quoi est dû l'échec de son interrogation. En fait, il peut relever de différents facteurs : formalisation de la requête par l'utilisateur, fonctionnement interne du MRI, contenu de la base de données...

Intégration d'un générateur à un SRI

Nous considérons ici qu'un SRI est constitué de quatre éléments : un moteur de recherche, une base de données, un analyseur de requête et un générateur. Ces éléments sont partagés en deux catégories de traitement : traitement de l'information (moteur de recherche et base de données) et traitement linguistique (analyseur et générateur). Une interface fait le lien entre l'utilisateur et le système de recherche d'information, et permet l'échange d'informations entre les deux.

Le processus d'interaction se décompose en deux étapes : de l'utilisateur vers le système et du système vers l'utilisateur. Nous présentons ici le rôle de notre générateur dans chacune de ces étapes.

De l'utilisateur vers le système : formulation de la requête

Comme nous l'avons déjà mentionné, une requête peut ne pas être adaptée au PIN et au moteur de recherche. Dans une telle situation, le système peut fonctionner de manière intelligente (transparence du processus de RI) ou encore de manière coopérative (interaction avec l'utilisateur), pour aider à la formulation et à la

reformulation de la requête dans le but d'obtenir de meilleurs résultats. Le générateur, en produisant des paraphrases de la requête initiale, fournit à l'utilisateur toutes les verbalisations possibles de son besoin d'information. Ceci ne peut qu'avoir un impact sur la perception et la formulation de la demande par l'utilisateur.

Nous abordons alors la génération sous un angle un peu particulier et différent de l'approche classique. Le destinataire final du message généré est en fait un système et non pas un humain. Le rôle de l'utilisateur est alors de sélectionner les paraphrases à transmettre au moteur de recherche. Nous posons plusieurs hypothèses sur cette étape de paraphrasage.

La première porte sur le mode d'interrogation qui doit être en langue naturelle. En effet, si le moteur ne traite pas la langue naturelle, il est inutile générer automatiquement des requêtes sous forme textuelle pour les lui fournir.

La seconde hypothèse est que toutes les requêtes sont paraphrasées par le générateur même si elles sont compréhensibles et aboutissent à des résultats pertinents. Le générateur utilise des critères syntaxiques (correction syntaxique, modification ou simplification de la requête par exemple). Il fait aussi usage de quelques connaissances particulières sur le moteur de recherche telles que les mots clés de l'application. Les paraphrases doivent respecter le besoin exprimé par l'utilisateur. C'est pourquoi la méthode que nous adoptons pour préserver ce besoin est de limiter le paraphrasage de la requête initiale à deux types de transformations : syntaxiques et lexicales (Ben Ali, Timimi 1999). De plus, le procédé de recherche d'information étant souvent basé sur l'appariement de termes, il semble pertinent de proposer plusieurs versions d'une même requête sans en modifier le sens afin d'augmenter le nombre de réponses potentielles.

Transformations syntaxiques

Cette reformulation consiste à transformer toute la structure syntaxique de la requête. **Une requête peut être restructurée, par insertion ou effacement de mots vides de sens ou par permutation des termes, en une nouvelle requête qui a une syntaxe très différente.** Cette transformation préserve ainsi les fonctions des

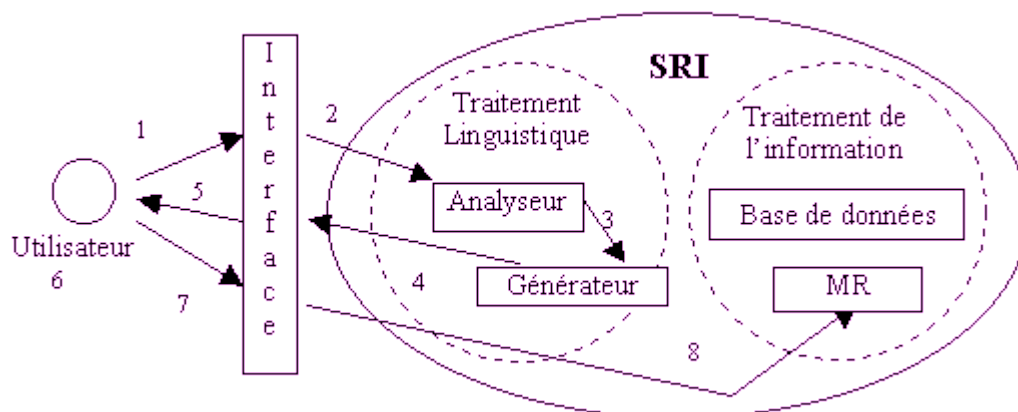
constituants et le contenu informatif. C'est le cas des requêtes suivantes : « *Augmentation de la consultation du Web ?* », « *La consultation du Web est-elle en augmentation ?* », « *La consultation du Web augmente-t-elle ?* ».

Notons également que le changement de voix de la requête (actif/passif) constitue une transformation souvent judicieuse.

Transformations lexicales

Les transformations paraphrastiques peuvent aussi être d'ordre lexical. Elles ne sont efficaces que si les mots utilisés sont recensés dans la base textuelle interrogée. Un fichier index par exemple peut contribuer à l'amélioration de la reformulation de requêtes. Il est alors possible d'employer les mots appropriés pour le paraphrasage des requêtes, particulièrement quand il s'agit de synonymie ou de mots à sens proche. Ceci nous confère une plus grande maîtrise des risques de modification du sens de la requête initiale. Les résultats obtenus suite à de telles transformations sont du type : « *Augmentation de la consultation du Web ?* », « *La consultation du Web est-elle en hausse ?* », « *La fréquentation de la toile augmente-t-elle ?* ».

Le générateur propose toutes les paraphrases possibles à l'utilisateur, qui choisit alors parmi ces possibilités celle(s) qui correspond(ent) le mieux à son besoin. Il peut sélectionner une ou plusieurs paraphrase(s) ainsi que sa requête initiale pour les soumettre par la suite au traitement du moteur de recherche. Ces phases sont représentées dans le schéma ci-dessous :



Ce processus est composé de huit étapes :

1. L'utilisateur exprime son besoin d'information en langue naturelle puis l'envoie à l'interface du système ;
2. L'interface achemine la demande à l'analyseur ;
3. La requête est analysée et la représentation formelle de son contenu est fournie en entrée au générateur ;
4. Le générateur produit des paraphrases de la requête et les renvoie à l'interface ;
5. L'interface propose les paraphrases du générateur à l'utilisateur ;
6. L'utilisateur sélectionne une ou plusieurs paraphrase(s), incluant ou non sa requête initiale ;
7. La sélection est retournée à l'interface ;
8. L'interface soumet la sélection de requêtes au moteur de recherche.

Du système vers l'utilisateur : donner des résultats

Dans cette seconde étape, l'approche de la génération est plus classique. Le but est de transmettre des éléments d'information à l'utilisateur. Comme pour la première étape, nous considérons que tout le flux d'information passe par le générateur avant d'être fourni à l'utilisateur final. Dans notre application, le générateur peut communiquer deux types d'informations.

Tout d'abord, il semble intéressant de fournir des éléments d'explication sur le fonctionnement du système de recherche d'information. Connaissant ce fonctionnement, l'utilisateur devrait être capable d'exprimer sa requête de façon mieux adaptée. L'objectif est que le système donne des réponses plus appropriées que « pas de document trouvé ». Il peut être possible d'expliquer à l'utilisateur quel est le domaine de la base, comment s'est effectuée la recherche, etc.

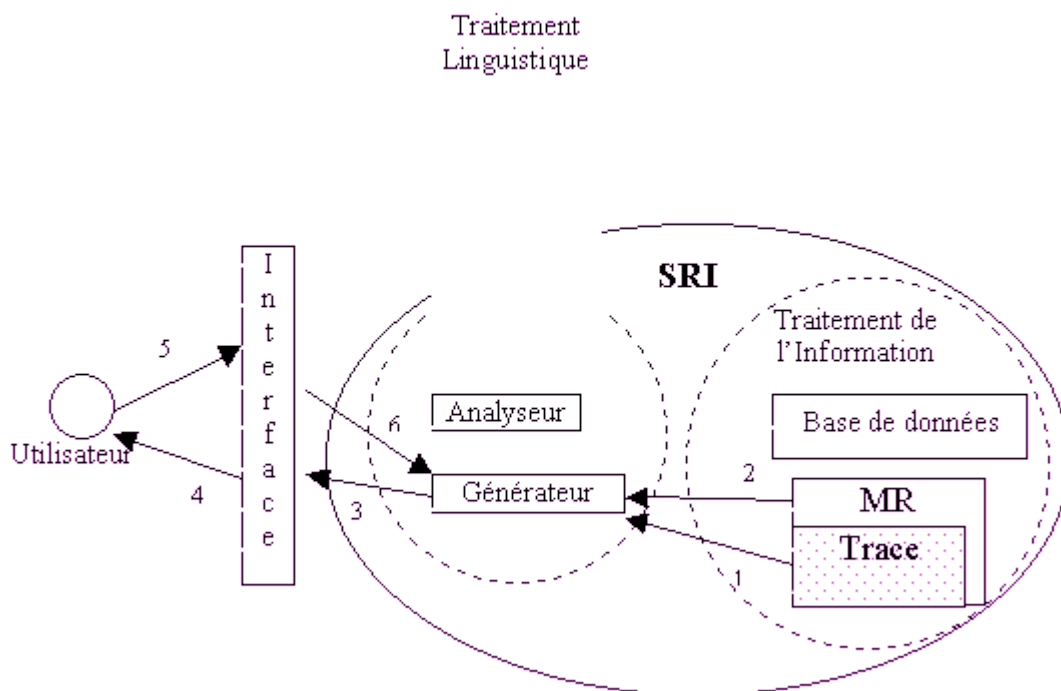
De plus, si le SRI contient par exemple un thesaurus, il est possible de fournir quelques indications d'extensions de recherche.

Le travail du générateur est donc basé sur une trace du processus de recherche qui constitue l'entrée de la production automatique en langue naturelle.

Le générateur peut également intervenir dans la présentation des résultats, fournis

généralement sous forme de liste. Il a la possibilité d'améliorer la présentation en structurant, triant, ou même résumant les réponses du SRI. L'utilisateur est alors en mesure de choisir l'affichage le plus approprié. Dans ce domaine, une étude sur la génération de commentaires de résultats bibliographiques a déjà été menée.

Ces interactions sont représentées sur le schéma suivant :



Les six étapes présentées sont les suivantes :

1. La trace est donnée en entrée de la tâche de génération ;
2. Dans le même temps, les résultats de la requête sont également fournis au générateur ;
3. Après le traitement, le générateur transmet des données (exprimées en langue naturelle) à l'interface ;
4. L'interface envoie les résultats à l'utilisateur ;
5. L'utilisateur peut demander d'autres affichages à l'interface ;
6. L'interface interroge le générateur selon les demandes de l'utilisateur.

Les étapes 5, 6, 3 et 4 sont répétées tant que l'utilisateur souhaite des présentations différentes.

L'intégration du générateur au SRI permet ainsi de faciliter les formulations de textes, qu'ils proviennent de l'utilisateur ou du moteur de recherche. Dans le premier cas, le générateur assiste l'utilisateur dans l'expression de sa requête ; dans le second, il améliore la compréhension de l'humain en lui fournissant des explications et des présentations de résultats plus adaptées.

Conclusion

Le générateur a été élaboré de manière à pouvoir traiter toute application. La première, brièvement décrite ici, nous a permis de tester cet outil et d'en améliorer les productions. Une collaboration entre différents pôles de recherche (recherche d'information, génération automatique de textes et analyse) a été à l'origine de la deuxième application, plus complexe, qui est en cours de développement. Associé à un SRI, le générateur peut aider les utilisateurs dans leur démarche, en facilitant l'interaction personne-système durant un processus de recherche d'information. En effet, cette tâche peut s'avérer fastidieuse et complexe si le système interrogé fonctionne sur des techniques classiques (appariement, pondération, etc.), qui agissent uniquement au cœur du moteur de recherche. L'intérêt et les apports de l'utilisation de traitements linguistiques (génération et analyse) concernent ici deux aspects.

L'interface est concernée en premier lieu. En effet, le recours à ces techniques augmente le degré de liberté de l'utilisateur en l'autorisant à verbaliser (formuler en langue naturelle) sa requête librement, c'est-à-dire sans contrainte de formalisation imposée par le système. Via une interface en langue naturelle un utilisateur peut exprimer au mieux son besoin en information.

Deuxièmement, si les techniques statistiques, par exemple, recherchent au niveau d'appariement une similarité parfaite entre plusieurs formes de surface (requête et contenu d'une base textuelle), les techniques linguistiques, quant à elles, autorisent en plus une ressemblance au niveau du contenu. Moyennant des phases d'analyse

diverses (morphologique, syntaxique, sémantique et du discours) et appropriées à l'application, les outils linguistiques permettent de passer de la similarité de surface à la similarité des thèmes, des idées voire des concepts.

En ce sens, nous sommes persuadés de l'intérêt d'associer notre générateur à un SRI. En outre, concernant l'amélioration de la pertinence des recherches, les différentes techniques (classiques et linguistiques) ne sont pas concurrentes mais complémentaires.[...] »

Annexe 5

Vous comprendrez que dans cette partie, le sujet « je » ne me représente pas...

Elisabeth a interviewé Carole le 3 mars 2004 et voilà ce que donne l'article :

« [...] Dans les clubs ou les associations que je fréquente, on ne parle depuis quelque temps que du « *Web Sémantique* », nouveauté censée révolutionner le Web.. Je me suis donc inscrite à divers ateliers ou conférences [...] pour tenter d'y voir plus clair. Hélas ! Plus j'entendais d'explications, moins je comprenais quoi que ce soit. Pire, j'avais la désagréable impression que certains orateurs, soit cultivaient l'ésotérisme à dessein, soit ne maîtrisaient pas du tout leur sujet.

Je ne peux résister au plaisir de vous citer ceci :

« Le Web Sémantique [...] possède un caractère sémantique dans la mesure où il s'appuie essentiellement sur la vertu d'objectivation de l'informatique pour valoriser les expressions particulières en les réunissant au-delà de leurs spécificités au sein d'un système d'information unique. Réciproquement, tout message, dans son ambition de participation à la communication planétaire, doit s'alourdir d'une enveloppe abstraite [...], cette abstraction qui n'est qu'humaine pouvant être convertie en principe en potentiel d'universalité et de discrimination par le pouvoir objectivant informatique. »

C'est à hurler de rire, ne dirait-on pas un texte style *pipotron*, généré par javascript ?

Lisez encore ceci :

« Le Web Sémantique est une forme de Graal de la Communication, et je pèse mes mots. »

Bon. Ces mots bien pesés, je m'étais persuadée que le Web Sémantique était au mieux une technologie en train de se chercher, et au pire une secte d'allumés, lorsque j'ai croisé Carole et lui ai raconté mes malheurs. Pouvait-elle m'aider ?

Carole, le Web Sémantique, c'est quoi ?

Aujourd'hui, me répond Carole, le contenu du Web est fait pour des lecteurs humains, mais pas pour des ordinateurs.

Et alors ? C'est grave, docteur ?

Oui, c'est grave pour faire une recherche efficace. Comme il y a de plus en plus de pages Web, des milliards, et que le moteur de recherche ne comprend pas le sens des documents, sa recherche sera de moins en moins efficace.

Mais on peut mettre des métatags dans la page pour aider les moteurs !

Oui, les métatags, [mots-clés qu'on met en tête de la page HTML, qui ne s'affichent pas mais sont visibles pour les moteurs de recherche *ndlw*], sont certes une petite avancée, mais elle reste très limitée. L'idée du *Web Sémantique* c'est de permettre une recherche intelligente sur le Web, faite par des ordinateurs et **basée sur des définitions** qu'ils puissent « comprendre », des définitions données pour le monde entier.

Quel genre de définitions ? Comme dans un dictionnaire ?

Pas exactement : disons qu'il faut mettre à disposition un langage de recherche, qui complète le Web d'aujourd'hui, et qui rende son contenu intelligible par des applications différentes. Autrement dit, qui lui donne un **sens**, c'est la signification du mot « Sémantique ».

Ce langage, c'est XML ?

Non ; XML n'est qu'un langage de description de document. Il permet de structurer l'information que l'on fait apparaître. Il consiste en de simple *tags* qui délimitent des données. Mais il ne donne pas la signification de ses structures ! Et sans la définition des structures, seul celui qui a créé le document est capable de le comprendre. Alors c'est **RDF** (*Resource Definition Framework*) qui va permettre de définir les structures, et cela au moyen de propositions ou « triplets » de type sujet/verbe/complément. Par exemple, une définition du type « *une femme d'oncle est une tante* » sera indispensable pour rechercher dans un arbre généalogique...

RDF est donc un langage ?

Mmoui...si tu veux ; ou plutôt un mode de **stockage des définitions**, comparable à

une feuille de style CSS. Mais en beaucoup plus complexe, car basé sur la théorie des triplets : tout document sera en effet défini par les occurrences de ces trois données, *sujet verbe complément*, ou plus exactement *sujet, prédicat, objet* encore appelés *objet, attribut, valeur* [A(O,V)]. A chaque tag XML correspond un triplet dans RDF. Dans la représentation graphique de RDF, le sujet et l'objet sont représentés par des sortes d'ellipses appelée "noeuds", et le prédicat par une flèche qui va du sujet vers l'objet.

On dirait de l'Intelligence Artificielle appliquée au Web ?

Non, car la grande différence, c'est que l'IA est basée sur la centralisation des données dans une Base De Données, alors que dans le Web Sémantique, les données peuvent être n'importe où, l'agent intelligent va les prendre en se baladant partout, sans qu'on ait besoin de les stocker.

Comment est-ce possible ?

C'est que je ne t'ai pas encore tout dit ! En plus d'une structure (XML) et d'un langage définissant cette structure (RDF), le Web Sémantique se caractérise par le fait que chaque partie du « triplet » RDF possède un identifiant appelé URI (*Uniform Resource Identifier*) qui permet à l'agent de le repérer.

Ca a un rapport avec l'URL ?

Oui l'URL est un cas particulier de l'URI. Les URIs assurent que les concepts ne sont pas juste des mots dans un document, mais qu'ils sont attachés à une définition unique que tout le monde peut trouver sur le Web.

En tout cas, cela change tout, car ainsi on pourra vraiment travailler en *grids* (*Travail en « grappe »*; *i.e. Infrastructure mutualisée de petites machines plutôt qu'une grosse machine*), en *peer to peer*, c'est à dire d'ordinateur à ordinateur, si les définitions sont partagées par tous.

Pour résumer, dans le Web Sémantique on a donc un support, le XML, qui organise le document, puis le RDF qui définit la structure c'est-à-dire la signification des tags XML, et enfin un univers non centralisé, mondial, avec des grids et du *peer to peer*.

Et l'ontologie là dedans c'est quoi ? On en entend souvent parler à propos du Web Sémantique, or l'ontologie, c'est l'étude de l'être en tant qu'être, si je me souviens

de mes cours de philo !

Effectivement. Mais dans le cas du Web Sémantique, ce mot comme pas mal d'autres - dont "triplets" - est emprunté au vocabulaire de la **logique formelle** et signifie tout à fait autre chose ! Je ne voulais pas t'en parler pour ne pas trop compliquer les choses, mais puisque c'est toi qui pose la question... En mathématique informatique, donc, une ontologie c'est la spécification formelle de la représentation des concepts, objets et entités et de leurs relations.

Et ça sert à quoi ?

Une ontologie typique sur le Web a une *taxonomie* et un répertoire de *règles d'inférence* dans un domaine de connaissances donné. La taxonomie définit des classes d'objets et leurs relations, exactement comme le fait par exemple la classification des animaux en *embranchements, classes, ordres, familles, genres, espèces et sous espèces* . Les règles d'inférence peuvent apporter une plus grande visualisation des éléments. Une ontologie peut exprimer la règle « *si un code département est associé avec un code région et qu'une adresse utilise ce code département, alors cette adresse est associée avec le code région* ». Avec le langage permettant d'écrire les ontologies, OWL (*Web Ontology language*), et RDF, des applications vont pouvoir utiliser et traiter les informations contenues dans des documents, indépendamment des humains.

Quel genre d'applications ?

Par exemple dans le domaine du *commerce électronique*. Imaginons un consommateur qui veut acheter une voiture. Un agent intelligent va chercher à sa place toutes les voitures correspondant à ses critères, comparer les prix, vérifier la disponibilité, les délais de livraison etc. en fouillant dans les documents du Web décrits selon les standards du Web Sémantique. Autre exemple, donné par l'inventeur du Web Sémantique qui est d'ailleurs le même que celui du Web tout court, à savoir Sir Tim Berners-Lee : je reçois un appel sur mon téléphone, aussitôt il va transmettre à tous les appareils possédant un réglage de volume sonore, hi-fi, télé.. l'ordre de baisser le son pour que je puisse téléphoner en paix..

Mais cela suppose que les documents en question soient décrits comme le veut le Web Sémantique, et c'est loin d'être le cas !

Pas encore mais ça vient. Le W3C s'occupe de définir les standards. Il y a toujours un temps d'évangélisation ! Les premières définitions de Tim Berners-Lee concernant le Web Sémantique sont sorties en 2001, mais à l'époque personne n'a compris, c'était trop tôt. Maintenant il y a déjà des applications qui marchent, les gens se mettent à faire leurs ontologies dans différents domaines, et surtout on a compris à quel point une recherche décentralisée sur le Web, et non pas sur le mode client-serveur, est utile aujourd'hui.

C'est la mort annoncée des moteurs classiques ?

Hé oui.

[...] »